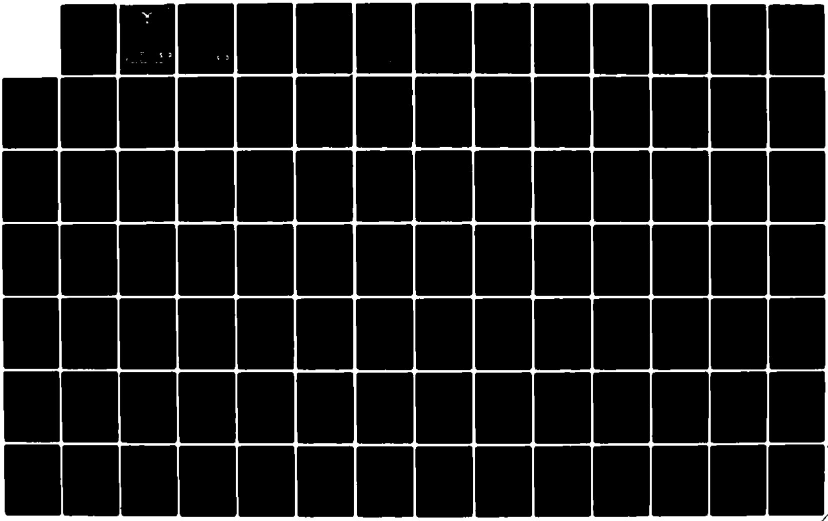
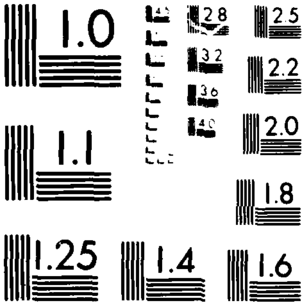
AD-A135 871 A SINE-WAVE GRATING CONTROLLER FOR VISION TESTING(U) 1/2

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL
OF ENGINEERING B D BAXLEY MAR 83 AFIT/GE/EE/83M-1

UNCLASSIFIED F/G 14/2 NL




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

EN/DDe

1

AD-A135871

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)

# AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/EE/83M-1

# A SINE-WAVE GRATING CONTROLLER FOR VISION TESTING

THESIS

AFIT/GE/EE/83M-1 Barry D. Baxley
Captain USAF

AFIT/GE/EE/83M-1

# A SINE-WAVE GRATING CONTROLLER FOR VISION TESTING

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

by

Barry D. Baxley
Captain USAF

Graduate Electrical Engineering

March 1983

## PREFACE

The Air Force Aerospace Medical Research Laboratory (AFAMRL) is conducting research into the development of a new means of testing the human visual system. The new eye test measures a subject's contrast sensitivity to sine-wave gratings on a video monitor. There is concern, however, that the brightness and contrast levels of the gratings displayed might vary during a test session from the settings made at the beginning of the session.

This thesis is the third effort aimed at developing a device which would automatically adjust brightness and contrast levels of a sine-wave grating in AFAMRL's vision test station. The first thesis examined the basic problem of how such a controller could be developed. A general design approach for the controller was proposed, and the hardware was ordered. The second thesis took the original design and built a prototype controller; however, most of the hardware and all the software in this prototype was completely untested.

In this thesis, most of the hardware and all the software were changed so that the controller could be used to monitor sine-wave gratings over a broader range of spatial frequencies than would have been possible with the original design.

I gratefully acknowledge the support of Major Arthur Ginsburg, Director, Aviation Vision Laboratory, AFAMRL as sponsor for this thesis.

My appreciation also goes to Mr Bob Durham, Mr Orville Wright, Capt Lee Baker and Mr Dan Zambon for all the invaluable technical assistance they provided during the design and debugging portions of this thesis.

I would especially like to thank Dr Matthew Kabrisky for the inspiration, guidance and support he provided as my advisor during this effort.

My deepest gratitude goes to my wife, Carin, and daughters, Mandy and Molly, for enduring the long absences, and providing moral support when I needed it most.

Barry D. Baxley

## CONTENTS

## CONTENTS (continued)

# LIST OF FIGURES

## LIST OF FIGURES (continued)

## LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AFAMRL | Air Force Aerospace Medical Research Laboratory |
| ABRT | Actual Brightness Value |
| AC | Alternating Current |
| ACON | Actual Contrast Value |
| ADC | Analog-to-Digital Converter |
| A/D | Analog-to-Digital |
| A* | Z80 Address Bus Line * |
| BCD | Binary-Coded Decimal |
| BDN | Brightness Down Control Line |
| BFCLR | Brightness Flag Clear Bit (OS0) |
| BNUFLG | Brightness Not Updated Flag Bit |
| BRTOUT | Brightness Output Value |
| BUFLG | Brightness Update Flag Bit (IS0) |
| BUSDIR | Bus Direction Line |
| CDN | Contrast Down Control Line |
| CFCLR | Contrast Flag Clear Bit (OS1) |
| CMOS | Complementary Metal Oxide Semiconductor |
| CONOUT | Contrast Output Value |
| CNUFLG | Contrast Not Updated Flag Bit |
| cpd | Cycles per Degree |
| CUFLG | Contrast Update Flag Bit (IS1) |
| DAC | Digital-to-Analog Converter |
| DBRT | Desired Brightness Value |
| DC | Direct Current |
| DCON | Desired Contrast Value |

## LIST OF ABBREVIATIONS (continued)

| | |
|---|---|
| DIP | Dual In-line Package |
| DRC | Dual-Rate Clock |
| DRDY | Data Ready Pulse |
| D/A | Digital-to-Analog |
| D* | Z80 Data Bus Line * |
| E-Beam | Electron Beam |
| EOC | End-of-Conversion Pulse |
| FF* | Flip-Flop * |
| fL | Foot-Lambert(s) |
| Hex | Hexadecimal |
| Hz | Hertz |
| IBPORT | Input Brightness Port (Input Port 1) |
| IC | Integrated Circuit |
| ICPORT | Input Contrast Port (Input Port 2) |
| IE* | Input Port * Enable Line |
| INREQ | Input Request Line |
| IS* | Input Status Bit * (Bit * of ISPORT) |
| ISPORT | Input Status Port (Input Port 0) |
| I/O | Input-Output |
| LED | Light-Emitting Diode |
| LAVG | Average Luminance Value |
| LMAX | Maximum Luminance Value |
| LMIN | Minimum Luminance Value |
| LSI | Large-Scale Integration |
| MDX-PIO | Mostek Parallel Input/Output Controller Board |

## LIST OF ABBREVIATIONS (continued)

| | |
|---|---|
| MHz | Megahertz |
| msec | Millisecond(s) |
| MSI | Medium-Scale Integration |
| nsec | Nanosecond(s) |
| OBPORT | Output Brightness Port (Output Port 1) |
| OCPORT | Output Contrast Port (Output Port 2) |
| OE* | Output Port * Enable Line |
| Op Amp | Operational Amplifier |
| OS* | Output Status Bit * (Bit * of OSPORT) |
| OSPORT | Output Status Port (Output Port 0) |
| OUTREQ | Output Request Line |
| PE | Preset Enable Control Line |
| RCLOCK | Reticon Clock |
| RDPORT | Reticon Data Port (Input Port 3) |
| RSCAN | Reticon Scan Bit (OS3) |
| SSPG | Scan Start-Pulse Generator |
| Synch | Synchronization |
| TTL | Transistor-Transistor Logic |
| UDBRT | Unscaled Desired Brightness |
| UDCON | Unscaled Desired Contrast |
| usec | Microseconds |
| WSC | WAIT-State Controller |
| WTBIT | WAIT Bit (OS3) |

# ABSTRACT

This report summarizes design, development and testing of a microprocessor-based system that controls brightness and contrast levels of video monitors. This controller is to be used by scientists in maintaining calibration of video monitors during vision research.

Prototype hardware for a video controller was designed and built. The controller requires a sine-wave grating to be displayed on a cathode ray tube (CRT). Luminance samples are measured directly from the CRT via a 512-element photodiode array. Analog information from the array is digitized and stored in a Z80 microcomputer. The sampled data are then used to compute CRT brightness and contrast. Computed brightness and contrast values are compared with desired values, and corrections are made to brightness and contrast control circuits for the video monitor.

Most of the hardware required for the controller was built and tested. Hardware completed included the video sampling and digitizing circuits, as well as Z80 input/output and user interface circuits. In addition, all of the controller software was written and tested.

Additional development is required before a working controller can be demonstrated. Brightness and contrast control circuits must be built and tested. Anomolies in performance of the commercially-procured video sampling circuits should be examined. Finally, closed-loop system operation must be tested and analyzed.

# I INTRODUCTION

## BACKGROUND

A considerable amount of current research has focused on developing an effective measure of human visual response. While the standard "eyechart" test is effective for measuring one's visual acuity (i.e. testing one's ability to see small, high contrast letters and symbols under bright light), it does not assess a subject's ability to detect or recognize a target in certain realistic (low-contrast, low-light) environments (Ref 1:4).

Consequently, the Air Force Aerospace Medical Research Laboratory (AFAMRL) is conducting research into new vision tests which would measure an individual's visual response over a broad range of spatial frequencies. One promising test approach involves measuring a subject's "contrast sensitivity" to sine-wave gratings over a range of spatial frequencies (Ref 1:6).

A sine-wave grating is defined as "a repeated sequence of light and dark bars that has a luminous profile which varies sinusoidally, about a mean luminance, with distance" (Ref 1:6). To the observer, these gratings look much like "fuzzy" bars.

In measuring a subject's contrast sensitivity, AFAMRL uses an optical signal generator to create a sine-wave grating on a video monitor. The subject views the monitor and adjusts the contrast of the grating until the bars are just at the threshold of becoming undetectable. Then, the test is repeated for several more bar widths (spatial frequencies). From this series of measurements, AFAMRL is able to determine the subject's contrast sensitivity over a spectrum of spatial frequencies.

A problem with the current test is that brightness and contrast of the test-pattern (grating) are adjusted only once during each test session, that being just before testing. As further adjustments during a test session might be disruptive to the subject, no further adjustments are made during the session. Researchers are concerned that luminance and contrast levels of the test pattern may vary during testing because of line voltage variations and calibration drift. Such variations induce errors in test measurements which could corrupt experimental results. To minimize these errors, AFAMRL has asked that a device be developed that will monitor and control luminance and contrast levels of the test patterns used in their experiments.

Capt Kenneth Martindale began development of the required controller as an AFIT thesis project in 1980. In his project, Martindale designed a controller based on a Z80 microprocessor, a Reticon Photodiode Array, and Datel Analog-to-Digital (A/D) and Digital-to-Analog (D/A) Converters. AFAMRL was satisfied with the preliminary design and consequently purchased the hardware needed to build the proposed controller. Unfortunately, the equipment arrived too late for Martindale to begin construction.

In 1981, Capt Albert Lawson resumed development of Martindale's controller. Lawson used Martindale's design and built most of the hardware for a controller. In addition, Lawson developed several Z80 assembly language programs for the controller. Because of the large scope of this effort, Lawson was unable to complete the controller development in the time allotted.

## PROBLEM STATEMENT

The objective of this thesis was to develop a system which could control contrast and brightness of a sine-wave grating on a video monitor. The system should be capable of monitoring and controlling gratings with spatial frequencies in the range of three to ten cycles per degree (cpd).

The original approach was to complete development and testing of the controller designed by Martindale and built by Lawson. This approach would have produced a controller capable of monitoring gratings over a relatively low spatial frequency range (about two to six cpd).

The original design could not have been adapted to control gratings with spatial frequencies higher than six cpd without a substantial number of hardware and software design changes. Detailed analysis of the original controller design, however, indicated that several hardware and software changes would be beneficial for other reasons (see Chapter II). Consequently, after two months of careful study, it was determined that the Martindale-Lawson controller would be redesigned.

## SCOPE

Because of the extensive amount of redesign required to meet the objective of controlling gratings over a broad range of spatial frequencies, it was not possible to complete the development as originally planned. However, most of the hardware and software in the new design were constructed and tested in this thesis project. Specifically, changes were made in the design of the following circuits:

1) Reticon Sampling and Digitizing circuits

2) Z80 Input/Output circuits

3) User-Interface circuits

Each of these circuits has been built and tested. In addition, the controller software has been completely revised. Like the hardware, the new software has been thoroughly tested, and correct operation has been verified. Specific reasons for each of the hardware and software changes will be detailed in Chapter II.

## ASSUMPTIONS

This thesis was based on several assumptions with regard to the Reticon hardware used. These assumptions are listed below.

1) The Reticon hardware was assumed to provide analog voltages proportional to the light intensity present on each of the Reticon array's photodiodes.

2) It was assumed that output "drift" on each of the photodiodes would be small (i.e. output signals for a single

photodiode would be consistent on both a sample-to-sample basis, as well as a day-to-day basis). For example, for a 20-foot-Lambert input on a specific photodiode, the Reticon hardware would produce a consistent output voltage. This voltage should not vary by more than a few percent.

3) It was also assumed that there would be consistency in output signals for all photodiodes given a uniform light input. Here, the assumption was that for a uniform input of, say, 20-foot-Lamberts (fL) on all of the photodiodes, the outputs corresponding to each of the photodiodes would be consistent to within a few percent.

4) Finally, it was assumed that the acceptance angle of each of the photodiodes would be narrow enough that a focusing lens would not be necessary when sampling the light from a video monitor's screen.

## GENERAL APPROACH

The first two months of this project involved careful analysis of the hardware and software developed by Martindale and Lawson. This task took much longer than it should have. Due to inadequate documentation, it was extremely difficult to decipher what Lawson had constructed, and to determine how it was supposed to operate. Almost all of Lawson's diagrams were basically wiring diagrams. There were no clear functional diagrams to explain how his circuits were supposed to operate. The text did not help much either. This task was further complicated by the complexity of Lawson's circuits. Lawson usually designed circuits using basic logic gates, when the same functions could have been performed better by MSI or LSI integrated circuits. Consequently, the task of understanding exactly what Lawson had constructed took a full two months.

After studying the existing hardware and software, it became obvious that most of the existing controller design should be changed. Chapter II details specific reasons for redesigning several components of the controller. The second phase of this project, therefore, involved redesigning several of the controller's circuits.

The third phase was hardware construction and debugging. Some time was lost in this phase due to hardware failures. Initially, the Reticon RL1024G photodiode array (used for sampling the light levels on the video monitor) failed and a replacement had to be obtained. Later, both Reticon circuit boards used with the array (an RG-100B "motherboard" and an RG-105 array board) were destroyed when a wire carrying 120 VAC was accidently dropped on the Reticon motherboard. A new array board was purchased to replace the one that was destroyed. All integrated circuits and several other components (capacitors, MOSFETs, etc.) were replaced on the motherboard. The Reticon circuits were finally operational after two months of repair. Aside from the hardware failures, this phase of development went well.

Unfortunately, there was not enough time to complete construction of all hardware for the controller. The circuits which were actually built and tested include the

Reticon sampling and digitizing circuits, the Z80 input/output (I/O) circuits, and the user interface circuits. The circuits for controlling brightness and contrast must still be constructed and tested before complete, closed-loop system performance can be evaluated.

After debugging most of the hardware, the controller software was designed. Some hardware (such as the I/O circuits) could not be fully tested until the software was ready. Each of the software routines was tested individually on the Mostek MDX-CPU2 computer. In addition, the software was also useful in testing some of the hardware (e.g. the A/D converter and I/O circuits). Finally, the routines were integrated and correct operation of all existing hardware and software was verified.

## SEQUENCE OF PRESENTATION

The remaining chapters of this thesis provide more detail into what was accomplished in this thesis and what must yet be accomplished before a working sine-wave grating controller is operational. Specifically, Chapter II provides additional details as to why many of the circuits and virtually all of the software were redesigned. Chapter III provides a general description of the various components of the controller, while Chapter IV describes how the completed controller will operate. Chapter V summarizes the progress made and problems encountered in this thesis. Specific problems still encountered with the commercially-procured Reticon circuits are detailed in Chapter VI.

Finally, Chapter VII provides final conclusions and recommendations.

# II REDESIGN

While Chapter I explained in general terms why several components of the controller were redesigned, specific reasons for th^ modifications were not presented. This chapter will provide more detail into reasons for altering the existing design.

The first circuits to be modified were the Reticon sampling and digitizing circuits. There were several reasons for these changes. These will be discussed in the next section.

## RETICON SAMPLING/DIGITIZING CIRCUITS

The first modifications came about because Martindale's design called for sampling every fifteenth diode (Ref 2:17, 3:6). This approach would have limited the range of spatial frequencies which could be calibrated to only two to six cycles per degree (cdp).

Since it was desirable to control gratings with spatial frequencies of more than six cpd, the design was changed so that every photodiode on the Reticon array was sampled. By sampling all diodes, gratings of much higher frequencies could be controlled than were originally possible. The new approach, however, necessitated a complete redesign of both the synchronization and the input/output circuits.

Modifications were also required in the circuits used to start a scan of the Reticon array. Martindale recognized a need to synchronize the array scans with the vertical synch (retrace) pulses from the video monitor (Ref 2:16-18). However, Lawson designed and built the system so that array scans were triggered by the Z80. There are three problems with this approach:

1) Successive readings of the diodes would not be consistent because there would be no way to regulate the number of times the video monitor's electron-beam (E-beam) swept across the array. The array would only be triggered when the Z80 was ready to begin the next scan. But the Z80 is loaded differently at different times, causing the delay between commanded Reticon scans to vary from scan to scan. Thus four E-beam sweeps might occur between one pair of Reticon scans, while five or six sweeps might occur between the next pair.

This variation is unacceptable, because the Reticon array integrates the amount of light detected between array scans. A difference in the number of E-beam sweeps would cause a difference in the amount of light integrated. Varying output voltages would then be produced by the Reticon circuits on successive Reticon scans, even though brightness would not actually have changed. Ultimately, the Z80 would "see" these apparent brightness variations and would try to correct for them.

2) A problem, similar to the one above, is that the Z80 might initiate an array scan while the E-beam is tracing through the Reticon's aperture. In this case, some of the photodiodes would see more E-beam traces than others in the same scan. The Z80 would interpret this as a change in both

brightness and contrast, and would again try to compensate.

3) A third problem with using the Z80 to trigger the Reticon circuits, is that time delays between scans of the Reticon array would be too long. A maximum integration time of 40 milliseconds (msec) is allowed between Reticon scans, before "dark current" becomes a significant error source (Ref 4:2). Lawson's software was relatively inefficient, and would easily have exceeded the 40-msec threshold.

A third change in the sampling/digitizing circuits was made to simplify the hardware and software. Martindale's design called for a 12-bit analog-to-digital converter (ADC) to process the Reticon samples. This degree of accuracy did not seem to be warranted when considering the error sources present in the sampling circuits. Although not specified in the product brochure, deviations of several millivolts are typically present in Reticon voltage samples (see Chapter VI). In addition, the human eye only has a relative contrast resolution accuracy of about five bits (Ref 5). Thus maintaining 12 bits of accuracy in measuring video luminance levels was deemed unnecessary. This design change led to substantial simplifications in both hardware and software, since it was no longer necessary to manipulate and process 12-bit data words.

A fourth change in the sampling circuits had to be made because of a device failure. The original design called for a Reticon RL1024G photodiode array. This chip contained a linear array of 1024 photodiodes on a one-inch semiconductor substrate. In July 1982, this chip was found to have failed. Because of production problems at Reticon, a replacement for the RL1024G would not have been available for ten weeks or longer. It was then determined that a Reticon RL512G (containing 512 photodiodes on a half-inch substrate) would be an acceptable replacement. The new chip and its corresponding (RC105) array board were readily available, and the RL512G cost only $300 (half the cost of the RL1024G).

## INPUT/OUTPUT CIRCUIT REDESIGN

Once the Reticon sampling and digitizing circuits were redesigned, it became necessary to change the Z80 input/output (I/O) circuits as well. The new design called for sampling each of the Reticon array's 512 photodiodes. Consequently, relatively high data-input rates became necessary, so that all of these samples could be read into the computer in the time allowed. The Mostek Parallel Input/Output (MDX-PIO) Controller, used in the Martindale-Lawson design, was simply not fast enough for this task.

The I/O circuits were redesigned to maximize the rate at which data could be read. The upper limit on the rate at which data can be input to the Z80 is limited by the Z80 clock. Using the Z80's block input command, INIR, samples can be read every 21 clock cycles. For the 2.5 megahertz (MHz) Z80 used in the Mostek MDX-CPU2 card, it is possible to read samples in as little as 8.4 microseconds (usecs). At this rate, all 512 photodiodes on the Reticon array could conceivably be read in about 4.3 milliseconds (msecs). Since the video monitor's electron beam (E-beam) scans across the array every 16.7 msecs (Ref 3:6), this data-input rate would easily permit sampling the entire array before the next E-beam traced through the array.

The I/O circuits were, therefore, redesigned to permit data to be read directly into the Z80, without the use of the MDX-PIO Controller. Other changes in the I/O circuits also came about as the various other circuits changed. The new circuits used different control signals ("flags") than those used in Lawson's design. A detailed description of the new I/O circuits is presented in Chapter IV.

## USER-INTERFACE CIRCUIT REDESIGN

The user-interface circuits were redesigned to make the controller more "user-friendly". Lawson's design for entering data worked, but it was difficult to use. To change a value for desired contrast, for example, the user might have to toggle switches more than 50 times. In addition, Lawson's circuits often "woke up" in illegal states when the system was first powered up. Features of the new design are summarized below.

1) With the new design, the user can easily change values for desired brightness and contrast by toggling switches up or down. A dual-rate clock in these circuits increments or decrements the values displayed quickly. The user does not have to repetitively toggle a switch up and down. Also, if the user passes the desired value, he need

only push the SET switch in the opposite direction to reverse the direction of counting.

2) Default values for brightness or contrast can be set up so the system always resets to the default values when the Master Reset switch is pressed.

3) Values for desired brightness are entered in foot-Lamberts (fL) (valid range is 0 to 39 fL, in unit increments).

4) Values for desired contrast are entered in percent (valid range is 0 to 99 %, in unit increments).

5) Desired values for brightness and contrast can be updated in either of two modes: continuous update, or manual update. In the continuous mode, screen brightness and/or contrast are updated continuously as the displayed values are changed. In the manual mode, new values for brightness and/or contrast can be set up while the screen values are left unchanged. When the operator is ready to update the screen, he need only toggle a switch. For a more detailed description of this feature, see Chapter III.

In addition to the user enhancements described above, the user-interface circuits were also changed so that the internal logic could be more easily understood. Lawson's design used discrete gates (i.e. AND's, OR's, JK-Flip Flops, etc.) to implement counting and reset functions. While this design worked, it was functionally complex and would be difficult to troubleshoot in case of failure. The new design uses MSI counters to perform the counting and reset functions. These counters also provide the benefit of being programmable. They can be preset to desired values so that the controller doesn't "wake-up" in an undesirable or illegal state.

## SOFTWARE REDESIGN

The software developed by Lawson was changed for several reasons. First, it was completely untested. Lawson developed the software to the point that it compiled properly and left it at that. Lawson's software was also inappropriate for the redesigned hardware. Input/output hardware, status flags, even data word-lengths were all different. Consequently, Lawson's software was finally abandoned completely.

The new software was designed with speed of execution in mind at all times. A detailed description of the new routines is provided in Appendix B.

## SUMMARY

This chapter has provided some background into specific reasons for redesigning individual components of the Martindale-Lawson circuits. The next chapter will describe, in general terms, what the various elements of the current controller are and how they operate.

## III CONTROLLER HARDWARE

### SYSTEM OVERVIEW

Figure 1 illustrates the general concept of how the various subsystems fit together to form the overall controller. An Optronix Generator sends a video signal to a monitor, where a sine-wave grating is produced. A Reticon photodiode array is mounted on the monitor's screen. This array and its associated sampling circuits scan the monitor and produce an analog signal representing the luminance detected on each of the array's 512 photodiodes. This

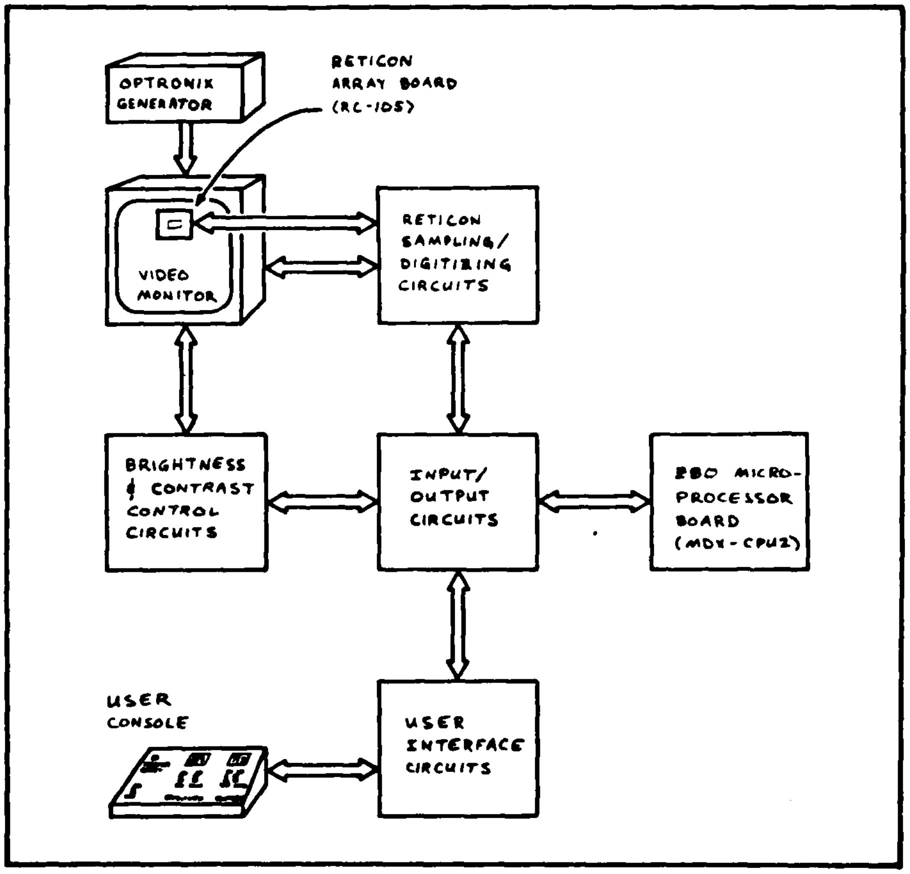

OPTRONIX GENERATOR
RETICON ARRAY BOARD (RC-105)
VIDEO MONITOR
RETICON SAMPLING/ DIGITIZING CIRCUITS
BRIGHTNESS & CONTRAST CONTROL CIRCUITS
INPUT/ OUTPUT CIRCUITS
Z80 MICRO-PROCESSOR BOARD (MDX-CPU2)
USER CONSOLE
USER INTERFACE CIRCUITS

Figure 1. Controller System Components

string of samples is digitized and sent to the Z80 processor via the Input/Output (I/O) circuits.

The Z80 analyzes the Reticon data and computes the actual brightness and contrast present on the video monitor. These values are compared with saved values for desired brightness and contrast. If the desired ana actual values differ, corrections are then sent to the circuits which control the monitor's brightness and contrast.

The system console allows the user to reset the system to preset values for desired brightness and contrast or to set up new values. In addition, a switch is provided on the console to allow the operator to switch the luminance controller out of the vision testing system altogether, if desired.

The following sections describe each of the circuits illustrated in Figure 1 in more detail.

## RETICON SAMPLING/DIGITIZING CIRCUITS

A functional diagram of the Reticon sampling/digitizing circuits is depicted in Figure 2. As described above, the Optronix Generator produces a sine-wave grating on the video monitor's screen. Vertical "synch" pulses from the monitor trigger a Scan Start-Pulse Generator (SSPG). This circuit synchronizes the vertical synch pulses with the Reticon clock to produce a start pulse for the Reticon circuits. In addition, the SSPG provides a user-variable delay. This allows the user to set the time (relative to the vertical synch pulses) at which a "scan" of the monitor's screen is

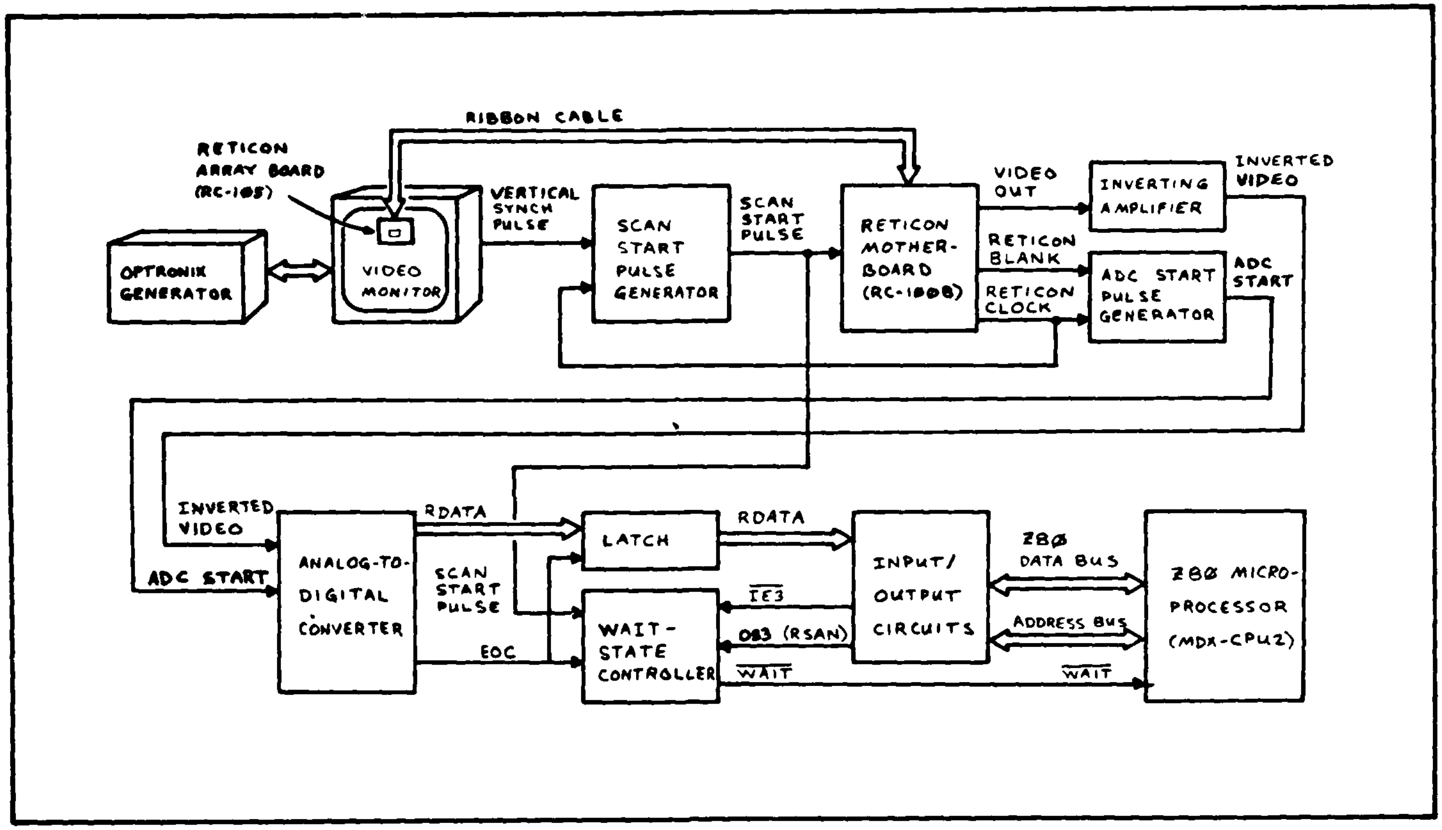


Fig 2. Reticon Sampling/Digitizing Circuits

to start. Thus the user can set the Reticon scans to occur in a "time window" during which the electron beam of the video monitor does not trace across the Reticon array.

The Reticon circuits include an RC100B "motherboard" and an RC105 array board. The array board is mounted on the video monitor's screen. When triggered by the start pulse, the Reticon circuits "scan" the monitor and produce a string of 512 samples. Each sample's voltage is proportional to the light detected by the corresponding photodiode on the Reticon array. These samples appear as a "boxcar" waveform at the video output of the Reticon motherboard.

The "video out" signal is then inverted by the Inverting Amplifier for digitization by the analog-to-digital converter (ADC) circuit. This inversion is necessary because the Reticon video output is a 0 to -5V analog signal and the ADC used accepts inputs in the 0 to +5V range. The Inverting Amplifier also provides gain-adjustment for the Reticon video signal. This feature will be used when calibrating the controller for specific light-intensity levels.

The ADC Start Pulse Generator uses the Reticon clock and blanking outputs to determine when each new Reticon sample is ready to be digitized.

After each conversion is complete, the ADC's end-of-conversion (EOC) pulse strobes an 8-bit latch. This stores the new value for input to the RDPORT (Reticon Data Port) while the ADC circuit digitizes the next sample. In addition, the EOC pulse signals the WAIT-State Controller that a new sample has been digitized.

The WAIT-State Controller (WSC) provides synchronization between the Reticon scanning/digitizing circuits and the Z80 microprocessor. Very simply, the WSC synchronizes the Z80's attempts to read a new video scan with the actual start of the next scan. The Reticon scan is triggered by vertical synch pulses from the video monitor. The WSC then sychronizes the Z80 with the Reticon circuits. In addition, this circuit forces the processor to wait until each new sample is ready to be read at the RDPORT. A detailed description of how the WAIT-State Controller works is provided in Appendix A.

## INPUT/OUTPUT (I/O) CIRCUITS

The I/O circuits provide an interface between the Z80 microprocessor and the rest of the luminance controller system. As shown in Figure 3, the I/O is composed of an address decoding circuit, a data bus buffer, four input ports, and three output ports. Address decoding for a fourth output port is provided, however, a fourth output port is not required for the current system design.

The address decoder selects the appropriate input or output port when the processor attempts to read from or write to one of the seven ports used. The decoder also controls the direction of data flow in the data bus buffer, depending on whether the Z80 is attempting to read or write data. In general, the buffer is set for passing data from the Z80 data bus to the ports at all times, except when the decoder detects that the processor is trying to read one of

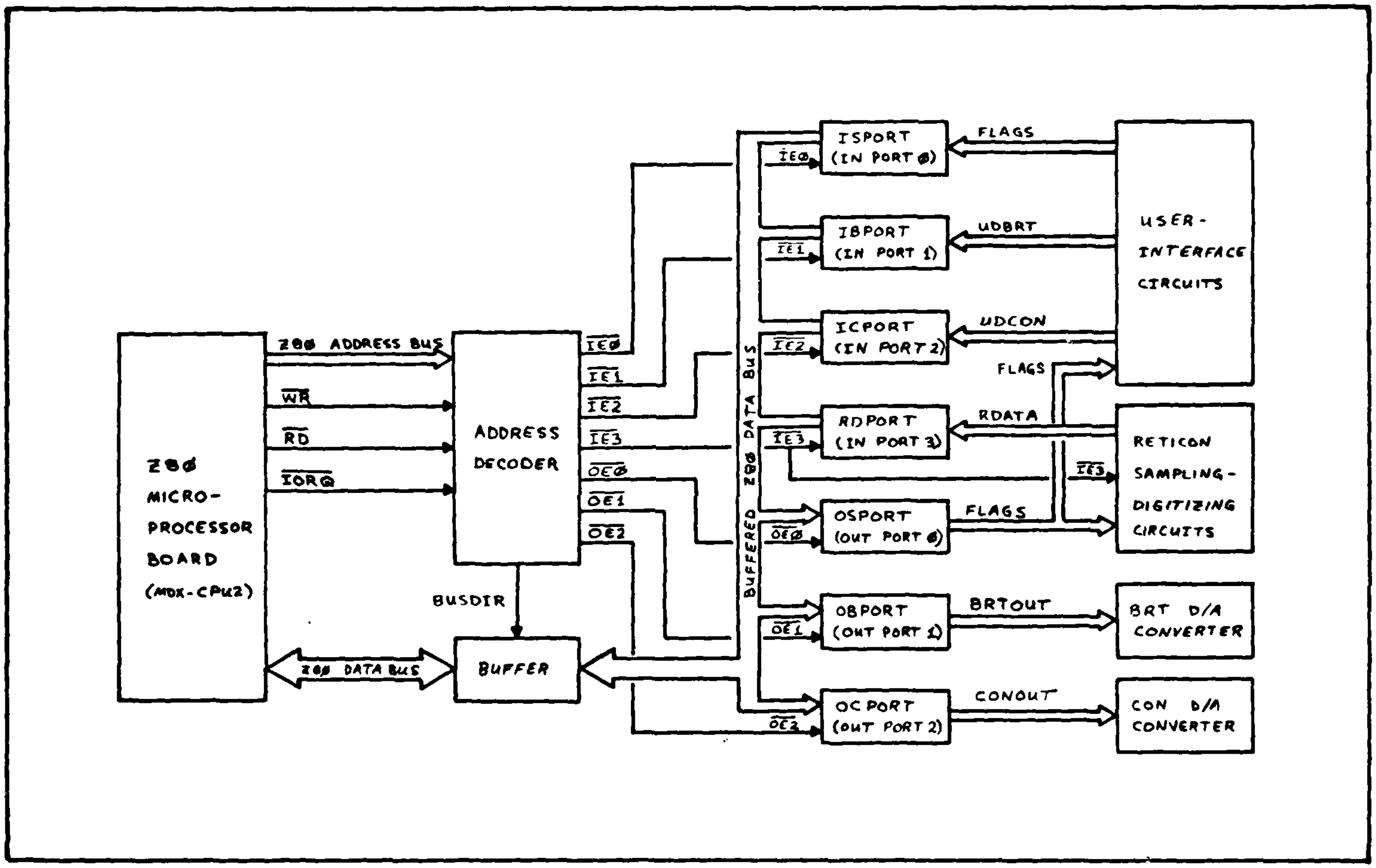


Fig 3. Input/Output Circuits

the four input ports.

The buffer is provided to minimize loading of the Z80 data bus. Due to fan-in, fan-out, and capacitive loading considerations, there are limits on the number of devices which can be tied to the data bus. The buffer allows several ports to be connected to the bus while placing only a single load on it. Actually, because the current luminance controller design places only seven ports on the data bus, the buffer could probably be omitted. However, the buffer was provided in case additional ports were added later.

The seven ports include the following:

1) Input port 0 is the input status port, or ISPORT. This port is used to input various status-related flags to the processor. At present, only bits 0 and 1 are used. Bit 0 (IS0) is the BUFLG (Brightness Update Flag) which signals that the user wishes to update the stored value for desired brightness. Bit 1 (IS1) is the CUFLG (Contrast Update Flag) that similarly indicates desired contrast is to be updated.

2) Input port 1 is the input brightness port, or IBPORT. This port passes the 8-bit binary-coded decimal (BCD) value for unscaled desired brightness (UDBRT) to the Z80 from the user interface circuits.

3) Input port 2 is the input contrast port (ICPORT) used to pass the 8-bit BCD value for unscaled desired contrast (UDCON) to the processor from the user interface circuits.

4) Input port 3 is the Reticon data port (RDPORT). This port sends the Reticon samples to the processor from the Reticon sampling/digitizing circuits.

5) Output port 0 is the output status port, or OSPORT. The Z80 sends control signals to the external hardware through this port. At present, only bits 0 to 3 are used. Bit 0 (OS0) is the brightness flag clear (BFCLR) bit which resets the brightness flags (BUFLG and BNUFLG) after desired brightness has been updated. Similarly, Bit 1 (OS1) is the contrast flag clear (CFCLR) bit which resets the contrast flags (CUFLG and CNUFLG) after desired contrast has been updated. Bit 2 (OS2) is the PRESET bit used to initialize the system to default values for desired brightness and contrast when the system first comes up or is reset. Bit 3 (OS3) is the Reticon scan (RSCAN) bit used by the Z80 to reset the Wait State Controller (see Appendix A).

6) Output port 1 is the output brightness port (OBPORT). This port sends brightness output (BRTOUT) values to the brightness control circuit to correct errors detected in luminance.

7) Output port 2 is the output contrast port (OCPORT). This port sends contrast output (CONOUT) values to the contrast control circuit when corrections in contrast level are needed.

## USER-INTERFACE CIRCUITS

Figure 4 is a functional diagram of the various components of the user-interface circuits. Switches on the

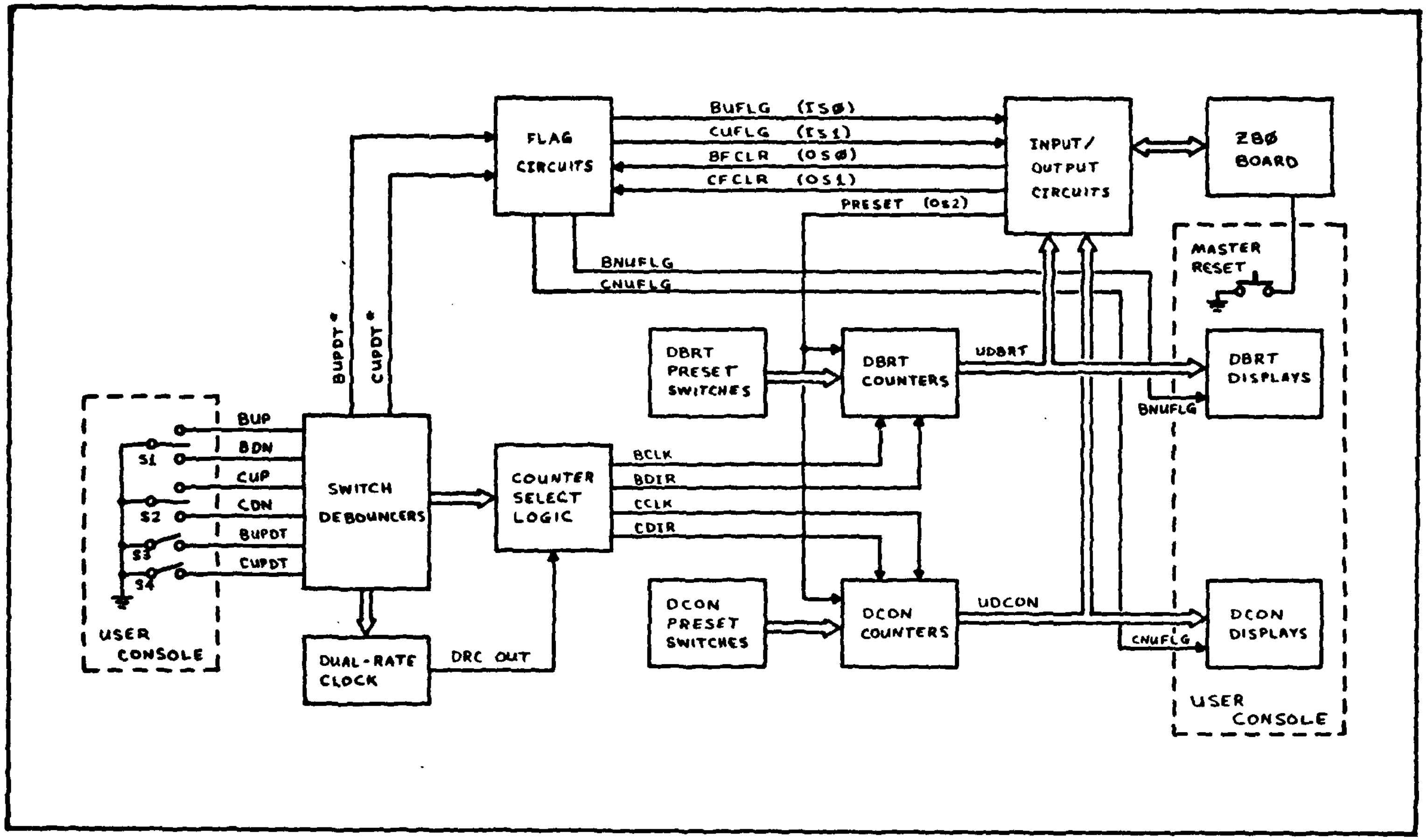


Fig 4. User-Interface Circuits Function Diagram

user console control setting and updating of desired values for brightness and contrast. These switches are debounced by an *MC14490 Hex Contact Bounce Eliminator*. Outputs from the switch debouncer drive a dual-rate clock, some counter select logic, and some flag circuits.

The dual-rate clock (DRC) produces a square-wave output whenever switch S1 or S2 is toggled up or down. This output is then sent to the counter select logic for driving the desired brightness and contrast counters. The clock rate is initially about one cycle per second when the operator first engages a switch. After four clock cycles the clock switches to a faster (4 Hertz) rate. The clock stops and resets as soon as the activating switch (S1 or S2) is released.

Outputs from the switch debouncers and the dual rate clock serve as inputs to the counter select logic, which increments or decrements the brightness and contrast counters (depending on which switch is activated, and whether the switch is lifted or depressed).

The brightness and contrast counters, in turn, send the brightness and contrast values, that have been selected, to displays and to the appropriate input ports. Other inputs to the counters include preset switches and a preset line. Together, these inputs provide default values for the counters whenever the controller is initialized (via the Master Reset button or the power on/off switch).

The flag circuits are used to signal the user when a value has been changed but not updated. In addition, these circuits signal the computer when brightness or contrast is to be updated. The I/O circuits, of course, provide the interface between the user-interface circuits and the Z80.

Although not shown in Figure 4, an important circuit, which should ultimately be added to the interface circuits, is a control/bypass switch. This switch would allow the user to switch the sine-wave grating controller completely out of the video testing system. This switch should be added when the monitor brightness and contrast control circuits are built.

## BRIGHTNESS AND CONTRAST CONTROL CIRCUITS

Another important component of the final controller consists of the brightness and contrast control circuits. Although not built or tested in this thesis, preliminary designs for these circuits are presented in Martindale's thesis (Ref 2:6-10, 2:22-25). Martindale bench tested the luminance processor and brightness control amplifier, and found that they provided adequate control over contrast and brightness.

Basically, these circuits send brightness and contrast control signals to the video monitor, as depicted in Figure 5. Control signals for brightness and contrast are sent from the Z80 to two Digital-to-Analog Converters (DAC's) via output ports 1 and 2. The output from the contrast DAC is input to a luminance processor circuit. This circuit controls contrast by regulating the monitor's video signal (inside the monitor). The output from the brightness DAC

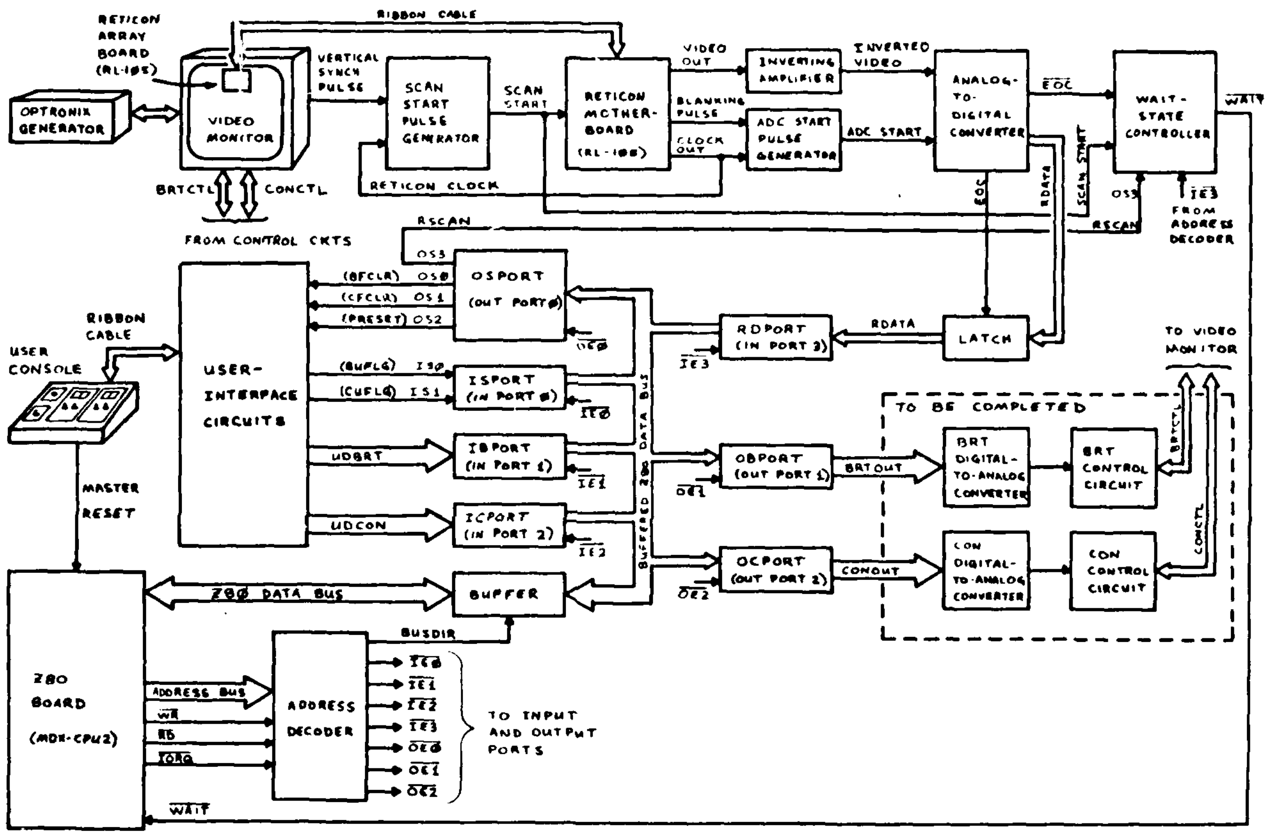


Fig 5. Controller System Function Diagram

drives a transistor which regulates the monitor's brightness (see Martindale's thesis, Ref 2:23).

## Z80 MICROPROCESSOR

The Z80 board is the heart of the controller. The board used in this thesis is a Mostek MDX-CPU2. The Z80 performs all of the overhead functions of the controller. Brightness samples from the Reticon sampling/digitizing circuits are read in via the I/O ports. Calculations are made from these samples to determine actual brightness and contrast values for the grating on the video monitor. The actual values are compared with desired values and correction signals are sent to the control circuits, if needed. The Z80 also updates desired values for brightness and contrast when the user signals (via the UPDATE switches) desired values are to be updated. The desired values are scaled by the Z80 for proper comparison with actual values. Detailed descriptions of the software routines which perform these functions are provided in Appendix B.

## SUMMARY

This chapter described the functions performed by the various components of the controller and explained how those components interface with each other. The next chapter will describe ultimate operation of the completed controller.

# IV SYSTEM OPERATION

This chapter describes the controller as it is currently envisioned.

The completed controller will occupy three separate enclosures, as depicted in Figure 6. One module must be mounted directly on the video monitor's screen. It does not, at present, appear practical to mount all the remaining controller hardware on the monitor's screen. Thus, a second enclosure, containing most of the remaining hardware, is mounted next to or on top of the monitor (the ribbon cable connecting the Reticon array board to the motherboard cannot be longer than 30 inches (Ref 4:1)). The third unit is the user console, with which the operator communicates with the controller. This is a separate unit, so the operator does not have to stand next to the monitor to operate the controller.

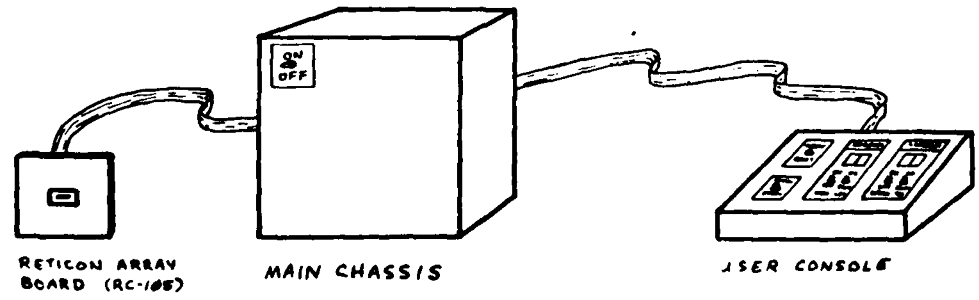


Figure 6. Controller Packaging

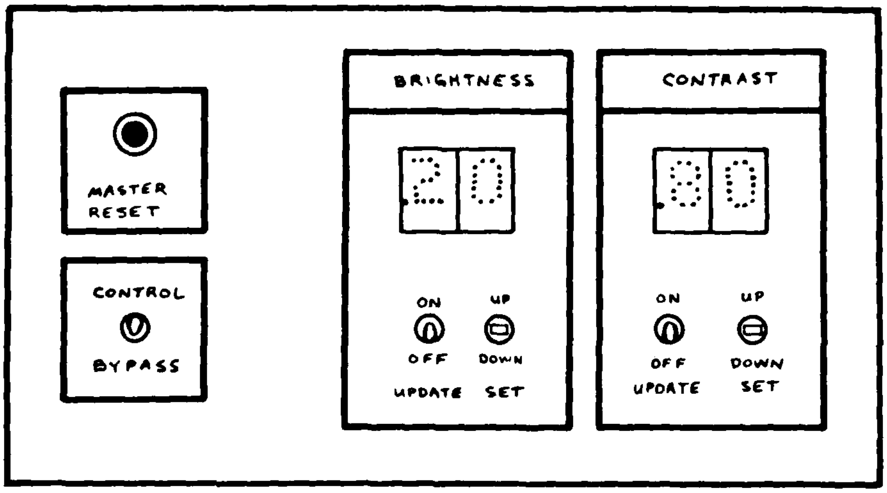


Figure 7. User Console

Since the console need only contain a few displays and switches, it is conceivable that the console could, in fact, be quite small. For example, it could easily be packaged in a case roughly the size and shape of a pocket calculator.

In normal operation, the user turns the system on by flipping the power switch on the control module. When the system is initialized, preset values for brightness and contrast are displayed automatically. The user then has the option of changing the values to new settings, if desired. In addition, the user can switch the controller in or out of the overall system via a switch on the console.

All switches and displays used by the operator, except the power (ON - OFF) switch, are mounted on the user console (Figure 7). The desired brightness setting is displayed on one pair of LED displays, while the contrast setting is displayed on a second pair. Switches below the displays control data entry. A Master Reset button is mounted on the console, as is a Control/Bypass switch (which is to be used for switching the controller in or out of the overall vision testing system).

Using the Brightness SET switch, brightness is variable over the range of 0 to 39 fL, in unit increments. For example, if brightness is to be increased from the current setting, the Brightness SET switch is lifted and released. By holding the switch in the "up" position, a dual-rate clock increments the setting automatically. The clock increments slowly (about one count per second) for four counts, then switches to a higher rate (about four counts per second) until the SET switch is released. This feature operates much like the SET button on many of today's digital watches or clocks. Similarly, if the SET switch is depressed (placed in the "down" position), the brightness setting is decremented in the same manner.

Thus, the user can quickly set brightness to any value between 0 and 39 fL. If he accidently overshoots the desired setting, he can simply toggle the switch in the opposite direction to correct the error.

Contrast is set much like brightness, except units and limits differ. Using the Contrast SET switch, contrast is variable between 0 and 99 percent in unit increments. The technique used in setting these values is as described above.

The controller was designed to allow the user to change desired settings in either of two modes. When the UPDATE switches are in the "ON" positions, screen brightness and contrast are updated automatically as the displayed settings are changed. When the UPDATE switches are in the "OFF" positions, screen brightness and contrast do not change. This feature allows the operator to set up a new value for desired brightness or contrast while the screen controls remain fixed at a prior setting. When the user wishes to update the screen to the new setting, he need only toggle the UPDATE switch on and off. As shown in Figure 7, brightness and contrast have independent UPDATE controls, so the operator has complete flexibility over how he wishes to change the desired settings.

The left-most decimal point on both of the displays signal that a setting has been changed, but not updated. For example, if the brightness setting is changed while the Brightness UPDATE switch is in the OFF position, the left decimal point (i.e. the "Brightness-Not-Updated" warning light) of the Brightness display would light up, indicating that actual brightness is still set at the previous setting. When the UPDATE switch is toggled on and off, the light goes out immediately and screen brightness is updated to the new setting. If the UPDATE switch is left in the ON position while the brightness setting changes, the warning light blinks on and off. This indicates that the value is being updated approximately every 40 milliseconds. As soon as the SET switch is released the light goes off. This shows that the value displayed is the current brightness setting.

The Contrast UPDATE switch works exactly like the Brightness UPDATE switch. As in the brightness display, the left decimal point indicates whether or not the displayed contrast setting has been updated.

## SUMMARY

This chapter provided a general description of how the completed controller will eventually operate. As noted in Chapter III, not all of the circuits to this end have been completed. The next chapter will describe construction and testing of those circuits which were completed in this thesis.

# V CONSTRUCTION AND TESTING

Several of the circuits described in Chapter III were breadboarded and tested in this thesis. This chapter will summarize the steps taken and problems encountered in testing and debugging those circuits and the driving software.

## RETICON SAMPLING/DIGITIZING CIRCUITS

As noted in Chapter II, the Reticon array was switched to an RL512G array early in this thesis. The original Reticon chip, an RL1024G array, was found to have failed before it was even tested in this thesis, although Lawson had tested the chip successfully in 1981. The cause of this failure was not clear; however, it appeared to have been caused by a manufacturing defect. Since the failure occurred over a year after the device was purchased (i.e. out-of-warranty), Reticon refused to provide a free replacement for the chip. It was also discovered, at this time, that, due to production problems at Reticon, a direct replacement would not be available until at least October, 1982. This would have caused unacceptable delays in development and testing for this thesis.

A Reticon RL512G array was then selected as an acceptable replacement for the RL1024G. The new device had basically the same geometry as the RL1024G. The only significant difference between the two was the number of photodiodes on the array. Whereas the original IC had 1024 photodiodes on a one-inch substrate, the RL512G had 512 photodiodes on a half-inch substrate.

The only problem in using the new chip in the sine-wave grating controller is that it increases the lowest possible spatial frequency that can be calibrated by the controller. The lowest frequency that could be controlled by the RL1024G was one cycle per inch (since at least one full cycle must be sampled by the array). This equates to about 2 cpd for the test configuration used at AFAMRL (the subject is seated about 3 meters from the video monitor). Using the RL512G array would double this lower limit to 4 cpd, since at least 2 cycles per inch would have to be present on the monitor. In light of the alternative of waiting 10 weeks or more for a new RL1024G array, the RL512G was considered an acceptable replacement. An advantage of the new chip was that the RL512G cost $300, whereas the RL1024G cost $600.

An RL512G array and its corresponding array board were purchased and tested. Using the Reticon's internal clock (set at 100 KHz) and triggering on the Reticon's internally generated start pulses, the Reticon output provided a reasonably good boxcar waveform.

Next, the Reticon circuits were tested using the external Reticon scan start circuit described in Appendix A. A Wavetek waveform generator, set at 60 Hz, was used to simulate the vertical synch pulses. Again, the Reticon circuit worked well.

## RETICON DIGITIZING AND INPUT/OUTPUT CIRCUITS

The ADC start pulse generator (described in Appendix A) was then built and tested. This circuit checked out properly, providing start pulses with proper delays and durations.

The inverting amplifier and ADC circuits were built next. Crude tests of these circuits (i.e. individual outputs of the ADC were monitored on an oscilloscope) indicated valid operation. However, these circuits were not thoroughly tested until the Z80 was connected to the system.

It was at this stage of development that the Reticon circuits were damaged when a wire carrying 120 VAC was inadvertently dropped on the Reticon motherboard. All ICs and several capacitors, MOSFETs, voltage regulators, etc. had to be replaced on the Reticon motherboard. The array board was finally replaced with a new board. Fortunately, the Reticon array and the analog-to-digital converter were not damaged in the accident. The Reticon circuits were eventually restored to seemingly normal operation (see Chapter VI).

After the Reticon circuits were repaired, the I/O circuits and WAIT-State Controller were completed. The SCAN routine was written and testing was begun. After some minor debugging, these circuits were shown to function properly. Complete Reticon scans were read into the Z80. When these circuits were all tested together, the ADC circuits were verified as well. Some undesirable effects were noted; but, these effects will be discussed in Chapter VI.

## USER-INTERFACE CIRCUITS

The interface circuits were built next. The original design for the dual-rate clock and counter circuits used TTL logic. After some minor debugging, these circuits counted up and down, as designed. But, when triggering the counters up or down by only a single digit, the counters consistently demonstrated a "glitch". For example, when counting down one digit at a time, the counter would alternately toggle up and down instead of down only. The same error occurred when counting up. Several attempts were made to overcome the problem, but eventually it became evident that the problem was an inherent problem with the 74192 and 74193 counters used. Eventually, the same basic design was followed using CMOS devices, instead of TTL. The new design worked perfectly.

The counter circuits were then connected to the I/O ports, together with the flag circuits. The software routines which interface with these circuits (i.e. early versions of STEST, BUPDAT, and CUPDAT) were then written and tested with the hardware. With only minor debugging, the hardware and software worked as intended.

The STEST routine, which calls BUPDAT and CUPDAT, was placed in a continuous loop and executed. Using the SET and UPDATE switches, values for brightness and contrast could be changed and updated exactly as described in Chapter IV. With the UPDATE switches in the OFF (down) position, the FLAG LED's on the brightness and contrast displays would remain off until one of the SET switches was triggered. The appropriate FLAG LED would then light and remain lit until reset by the corresponding UPDATE switch. A quick check of the RAM contents verified that the correct value was read.

With the UPDATE switches in the ON (up) position, the FLAG LED's would flicker on and off as the counter values were repetitively updated. The processor was again stopped and RAM contents verified that the latest displayed values were stored. All possible combinations of switch settings were tried and correct operation was similarly verified.

This marked the end of the hardware development conducted in this thesis. Several software routines were then written and tested.

## SOFTWARE TESTING

At this point, crude versions of the SCAN, STEST, BUPDAT, and CUPDAT routines were operational. The next step involved refining those routines already working and adding additional routines.

The first new routines tested were AVGBRT, SUMTBL and ADPAGE. Together, these programs compute the average of the Reticon data samples (i.e. the actual average brightness, or ABRT, measured by the Reticon array). With minor modifications, these routines were quickly debugged and verified. First, they were thoroughly tested using various combinations of known values. Later, these routines were tested together with the SCAN routine to compute actual averages of samples obtained from the Reticon array. As in their initial testing, these routines worked properly together.

Next, several routines were developed to compute actual contrast. The equation used in this computation was based on the Michelson definition of contrast (Ref 1:6):

$$C = \frac{Lmax - Lmin}{Lmax + Lmin} \tag{1}$$

Martindale (Ref 2:2) noted that

$$Lmax = Lavg + \Delta$$

and

$$Lmin = Lavg - \Delta$$

where

$$\Delta \equiv \text{Sine-Wave Grating Amplitude}$$

Thus

$$\begin{aligned} Lmax + Lmin &= (Lavg + \Delta) + (Lavg - \Delta) \\ &= 2\ Lavg \end{aligned} \tag{2}$$

The actual equation used in computing contrast is found by combining equations (1) and (2):

$$C = \frac{Lmax - Lmin}{2\ Lavg} \tag{3}$$

Other forms of this equation could also be used.

To use this equation, it was necessary to determine maximum and minimum sample values from the Reticon data. Two very similar routines, FMAX and FMIN, were then developed to sort through the data table and find those values. The average luminance value had already been computed by AVGBRT.

Once all variables were known, it was necessary to find a division routine which would compute contrast as in equation (3). It was noted that the largest possible divisor was 2 X 255 = 510. Thus a 9-bit divisor would be required. This would have called for a multiple precision division routine in the Z80 microprocessor. To simplify and speed up the division process, it was noted that the numerator could first be divided by Lavg, using a standard 16/8-bit division routine. This result could then be shifted right one bit to complete the computation.

Several existing "so-called" 16/8-bit division routines (Ref 6:190, 7:8-12, 8:135) were tried initially, but each of these routines failed whenever a dividend larger than 32,767 was used (i.e. whenever bit 15 of the dividend was set). By checking these routines against a division algorithm, a problem common to all of them was discovered. Each of the routines shifted the dividend left one bit, yet none of them tested to see if there was overflow in doing so. Thus, when bit 15 was initially set, this bit was always lost when shifting left. Apparently, none of the routines cited above was truely a 16/8-bit division routine.

Once this problem was recognized, it was possible to correct it. The actual algorithm used in the final DIVIDE routine was adapted from Leventhal (Ref 7:8-12). With the correction described in Appendix B, DIVIDE finally produced valid results for all possible 16-bit dividends.

With FMAX, FMIN, and DIVIDE working properly, CNTRST was written. It was noted that true contrast would have been a fraction, since contrast is defined in percent. Consequently, before computing actual contrast, ACON, from equation (3), the numerator was first multiplied by 256 (shifted left 8-bits), producing a full 16-bit dividend. ACON was then computed as a scaled integer value. CNTRST was finally tested and verified.

The values computed for actual brightness and contrast had to be compared with the desired values, so that corrections could be sent to the output circuits, if necessary. However, the values entered by the operator were originally stored in binary-coded decimal (BCD) form. Thus, before making a comparison, the user-entered values had to be scaled to a form which compared directly with the computed values.

This scaling was done by altering the original versions of BUPDAT and CUPDAT. In both routines, the user-entered (BCD) values are first converted to binary values by CONVRT. The binary forms are then multiplied by scale factors (see Appendix B). The results are saved as scaled values for desired brightness and contrast, DBRT and DCON. The new versions of BUPDAT and CUPDAT were tested and work properly.

Once the desired and computed values for contrast and brightness were in compatible forms, the values could be compared in correction routines. BADJ and CADJ were then written as potential correction routines, to be used when the output control circuits are completed. As noted in Appendix B, these routines are prototypes; routines which converge faster to steady-state values are certainly possible. The algorithms used in BADJ and CADJ were chosen for their simplicity and stability. Response from these routines should be overdamped. When the control circuits are constructed, performance tests can be conducted to choose between these and other potential routines.

SUMMARY

This chapter summarized the progress made in this thesis. As stated before, the hardware and software developed in this thesis have been thoroughly tested. With the exception of the commercially-procured Reticon circuits, everything seems to be working well. The next chapter will provide details of current problems with the Reticon circuits.

# VI CURRENT PROBLEMS

Although the hardware and software developed in this thesis have been tested successfully, there are some problems which must be resolved before an operational controller can be completed. This chapter summarizes those problems.

All of the problems center around the commercially-procured Reticon circuits. Some of the problems may be inherent in the Reticon circuits themselves, while others may be caused by circuit malfunctions in the Reticon motherboard. After the motherboard was damaged (see Chapter V), many of the components on the board were replaced. Some of the problems, described in this section, could be caused by a bad component on the motherboard.

The first problem noted is that the first two samples in each Reticon scan are not accurate. This problem appears to be an inherent function of the Reticon circuits, however, and not a problem which can, or even needs to be, corrected. The Reticon product brochure (Ref 4:4) provides specifications for several parameters, neglecting the first two, and last two, diodes. This indicates some error is inherent in those particular diodes. At present, there appears to be nothing wrong with the samples provided by the last two diodes. The first two, however, provide samples of consistently larger magnitudes (before being inverted by the inverting amplifier) than the remaining diodes. Because of limited slew rate in the inverting amplifier, the digitized sample from the first diode is generally corrupted further. This is not a real problem, however, because the errors in these samples are not substantial enough to affect the computed actual brightness, and the computation for actual contrast ignores the first two diode samples.

Another similar "glitch" appears sporadically in samples from diodes #350 to #354 (#15E to #162 Hex). These diodes intermittently produce samples of magnitudes as much as 0.25 Volts lower than diodes adjacent to them. The errors seen here are proportional to the magnitude of the samples, i.e. for low voltages, the error may be as small as a few millivolts. There is no known cause for this error. This error should not affect computed average brightness, because these samples would be averaged with samples from the remaining 500+ diodes. However, the error could affect the computed actual contrast, since this computation uses the minimum sample value, BMIN.

This error is, again, intermittent, and appears to be related to positioning of the ribbon cable connecting the array board to the motherboard. In certain positions, the problem goes away altogether. It is likely that this problem will vanish when the controller is properly packaged and mounted, since the problem seems to diminish when the array board is moved away from the motherboard.

Another problem of concern is that there appears to be a systematic non-uniformity in the samples produced by successive diodes on the array. Voltages present on the first several samples are consistently higher, by as much as 0.5 Volts, than those on later diodes, when a uniform amount of light is present on all diodes. This voltage deviation tapers off gradually from the few first diodes to about diode number #210 (#0D1 Hex), and is proportional to the size of samples produced (i.e. low voltages exhibit smaller offsets). This problem may be inherent in the Reticon circuits. Non-uniformity of Sensitivity is specified at ±9% by the Reticon product brochure (Ref 4:4). This could have an effect on computed values for both contrast and brightness. Performance tests should verify the magnitude of this problem.

Another significant problem is that the DC offset of Reticon samples shifts. When the array is darkened, the array output can be zeroed, as outlined in the alignment procedures for the Reticon circuits. Once this has been set, the darkened array's output should remain clamped at zero. However, the output shifts by as much as ±0.2 Volts over times as short as 3 or 4 minutes.

This offset deviation appears to be thermally-related, as it seems to get worse when the circuit has been running for a while. The problem may be caused by a defective MOSFET switch (there are four of these) or capacitor on the RC100B motherboard. This deviation was not noticed until after the accident that damaged the boards, therefore, it may be a result of something that was damaged at that time.

If not corrected, this problem could affect computed actual brightness and contrast. The offset would be detected as a shift in actual brightness, ABRT. ABRT is then used in the computation for actual contrast, as shown in Eq (3).

The last problem of concern is that an AC signal is sometimes present in the Reticon output. This looks like a 60 Hz signal riding on the boxcar waveform. The amplitude of the signal is proportional to the magnitude of the samples produced by the Reticon circuits. This problem just recently started. Cause of the problem is unknown.

## SUMMARY

As stated at the beginning of this chapter, all of the current problems center around the Reticon circuits. Some of these problems appear to be inherent in the Reticon circuits, themselves. Others may be caused by defective components on the RC100B motherboard. Some troubleshooting will be required in determining the cause of these problems. If not corrected, errors in computed actual brightness and contrast will result, and final closed-loop performance of the contıoller will be inadequate.

# VII CONCLUSIONS AND RECOMMENDATIONS

## CONCLUSIONS

Much of the original (Martindale-Lawson) design was changed in this thesis. In light of the increased range of spatial frequencies which may now be calibrated, the design simplifications made, and the improved user-interface, these modifications were more than justified.

All of the circuits and software, developed in this thesis, have been tested, and correct operation has been demonstrated.

There are still some problems to be worked out (Chapter VI), and several tasks which must still be completed before final closed-loop performance can be evaluated. It is expected, however, that the final controller will meet AFAMRL's needs as an effective tool for automatically calibrating the sine-wave gratings used in their vision testing.

## RECOMMENDATIONS

The problems, addressed in Chapter VI, must be resolved before a completed controller can be evaluated. The most serious problems appear to be correctable.

Once these problems are worked out, the output control circuits must be completed. Two digital-to-analog converter circuits, a luminance processor, and a brightness control amplifier must still be constructed, and a power supply must be purchased or constructed (at present, the hardware is powered by one of AFIT's laboratory power supplies).

In addition, it would be desirable to add some form of fault detection and warning system. This could be as simple as lighting an LED when actual brightness or contrast values differ from desired values by some tolerance. This feature would be very helpful to the user, in case one of the circuits in the controller failed.

Once the hardware is completed, the output control routines, BADJ and CADJ, must be tested and evaluated. These routines should work, but faster routines are possible. This task might involve comparing two or more routines, with a trade-off of speed versus stability.

Next, the controller must be calibrated and performance must be evaluated. For this task, calibration equipment (precision light sensors) will be needed from AFAMRL.

Finally, the completed controller must be packaged. A bracket will have to be constructed for mounting the Reticon array board on the video monitor. A small case must also be obtained for mounting the user console panel. A cabinet already exists for holding the remaining hardware. The new power supply and existing hardware must be mounted in that cabinet.

# BIBLIOGRAPHY

1. Ginsburg, Arthur P. Proposed New Vision Standards For the 1980's and Beyond: Contrast Sensitivity. AFAMRL-TR-80-121. Wright-Patterson AFB, Ohio, September, 1981.

2. Martindale, Kenneth L. Programmed Control of Optical Grating Scales for Visual Research. MS Thesis. Wright-Patterson AFB, Ohio, December 1980.

3. Lawson, Albert L. Programmed Control of a Sine-Wave Grating Generator for Visual Research. MS Thesis. Wright-Patterson AFB, Ohio, December 1981.

4. G Series Solid State Line Scanners, 128, 256, 512 and 1024 Elements. Product Brochure. Sunnyvale, California: Reticon Corporation, 1981.

5. Kabrisky, Matthew J. Professor of Electrical Engineering, Air Force Institute of Technology, (personal interview). Wright-Patterson AFB, Ohio, May 1982.

6. Barden, William, Jr. The Z-80 Microcomputer Handbook. Indianapolis, Indiana: Howard W. Sams & Company, Incorporated, 1980.

7. Leventhal, Lance A. Z80 Assembly Language Programming. Berkeley, California: OSBORNE/McGraw-Hill, 1979.

8. Zaks, Rodnay. Programming the Z80 (Third Edition). Sybex Incorporated, 1980.

9. Lancaster, Donald. TTL Cookbook. Indianapolis, Indiana: Howard W. Sams & Company, Incorporated, 1974.

10. Linear Applications Handbook. Santa Clara, California: National Semiconductor Corporation, 1978.

11. Engineering Product Handbook. Próduct Brochure. Mansfield, Massachussetts: Datel Systems, 1979.

12. Sargent, Murray and Shoemaker, Richard L. Interfacing Microcomputers to the Real World. Reading, Massachusetts: Addison-Wesley Publishing Company, 1981.

13. Wadsworth, Nat. Z80 Software Gourmet Guide & Cookbook. Rochelle Park, New Jersey: Hayden Book Company, Incorporated, 1979.

# APPENDIX A

# HARDWARE OPERATION

## USER-INTERFACE CIRCUITS

The data entry circuits consist of four debounced switches, a dual-rate clock circuit, four flag-generating flip-flops, two 2-decade counters, and four TIL-311 LED displays, as illustrated in Figure 8. These circuits were originally designed using standard TTL integrated circuits; however, when tested, the original circuits exhibited undesirable characteristics when attempting to increment or decrement the counters by a single digit (see discussion in Chapter V). After several unsuccessful attempts were made to correct the problem, the circuits were redesigned using CMOS IC's. The new design worked exactly as intended, so the TTL version was abandoned.

Desired values for brightness and contrast are entered via switches S1 - S4. Switches S1 and S2 are single-pole, 3-position, (On)-Off-(On) switches used for setting brightness and contrast values. Switches S3 and S4 are single-pole, 2-position, Off-On switches used for signaling that brightness or contrast values saved in the computer are to be updated.

As shown in Figure 8, all four switches are debounced by an MC14490 Hex Contact Bounce Eliminator. The debouncer outputs a logical "1" for an "open" input line, while a logical "0" is produced when an input line is "grounded". Thus when S1 and S2 are in their center "Off" position,

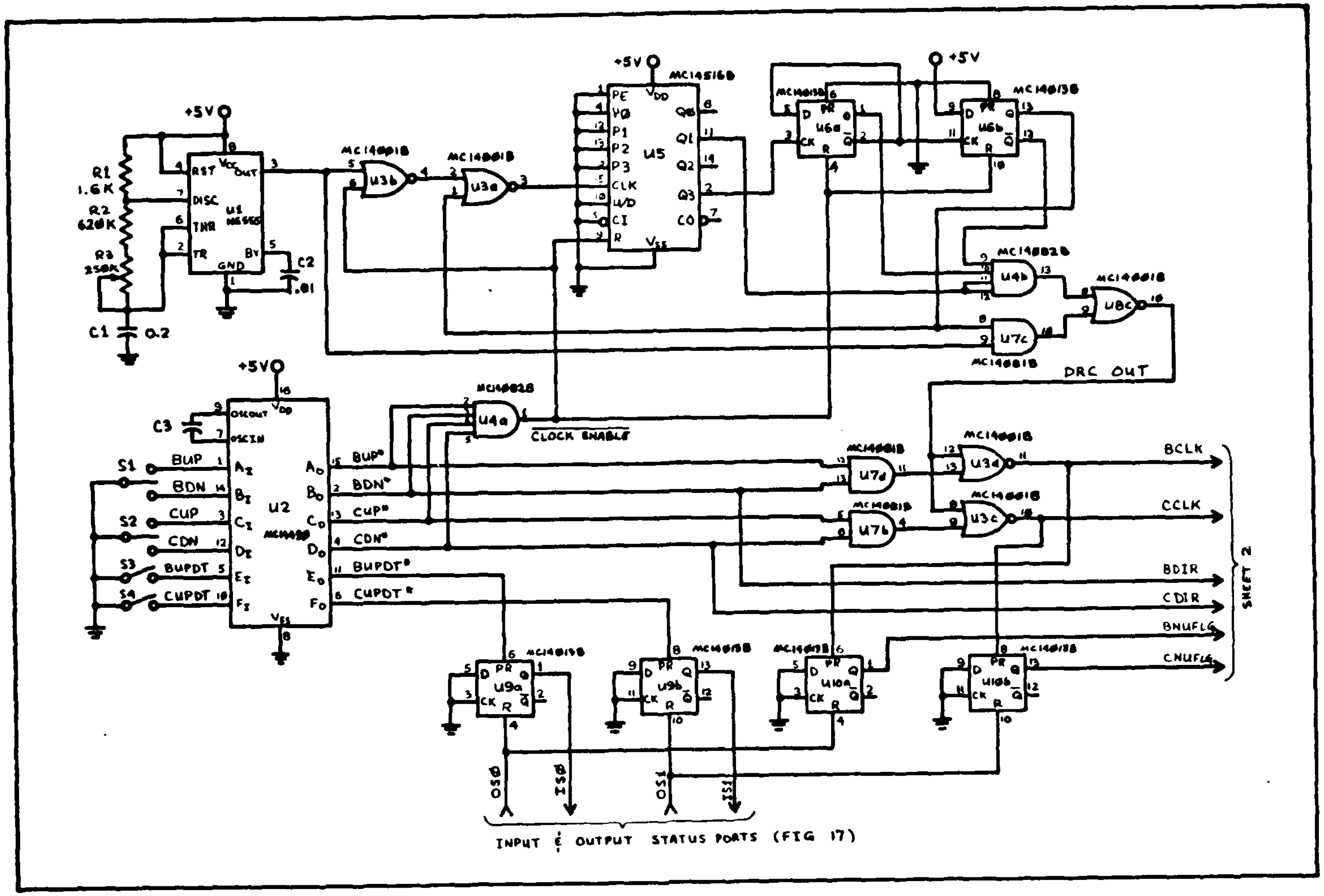


Fig 8. (Sheet 1) User-Interface Circuits Schematic

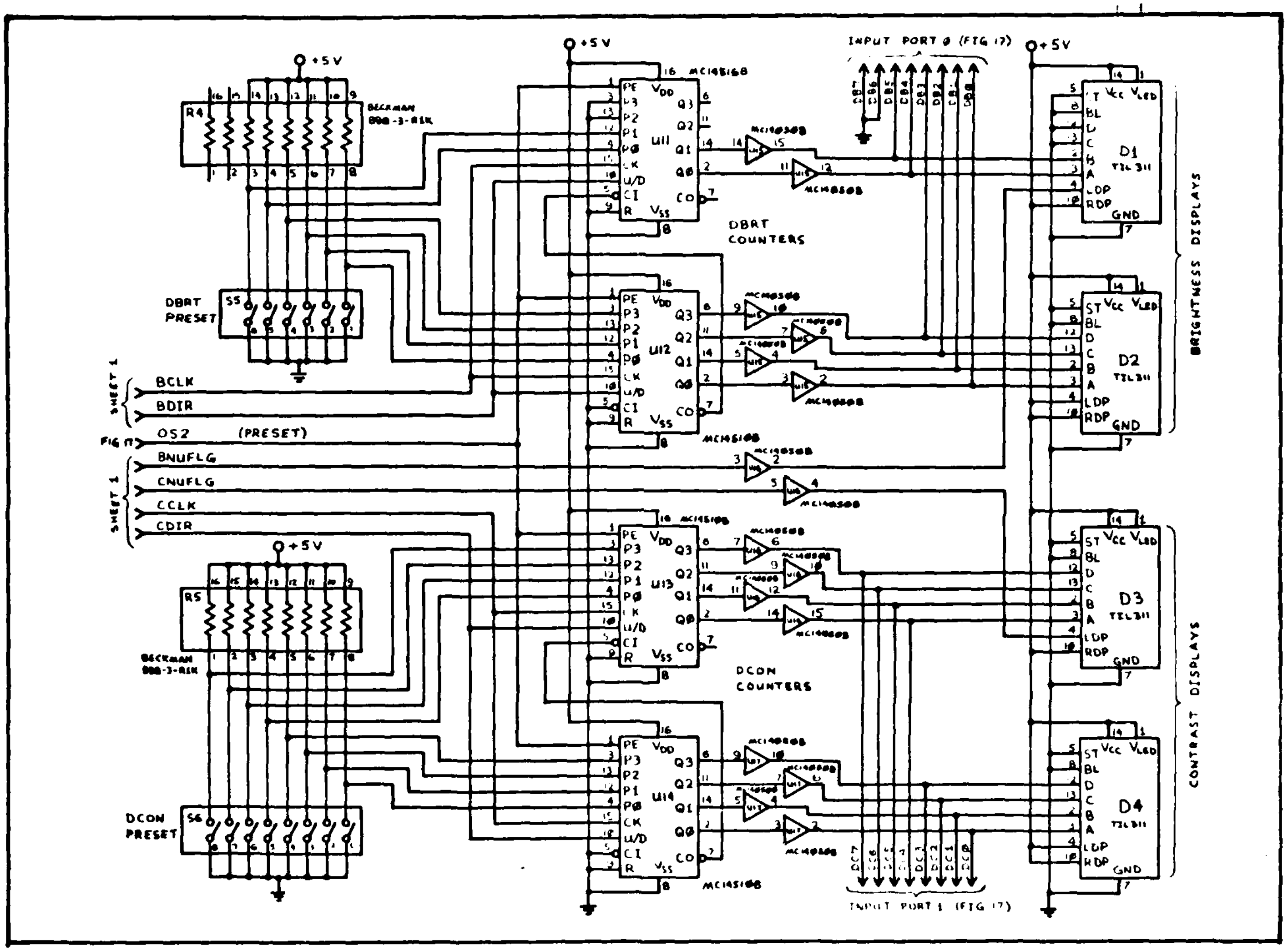


Fig 8. (Sheet 2) User-Interface Circuits Schematic

outputs A0, B0, C0 and D0 are all held high. Capacitor C3 sets the settling time for the bounce eliminators.

### Dual-Rate Clock Circuit

For clarity, the dual-rate clock (DRC) circuit is redrawn in Figure 9. The base frequency for the clock is set by a 555 timer, U1, which runs astably at approximately 4-Hertz. An MC14516 Binary Up/Down Counter, U5, divides the clock rate by 2, 4, 8 and 16 at its Q0, Q1, Q2, and Q3 outputs, respectively. An MC14013 Dual D Flip-Flop divides the Q3 output rate two more times. The counters are reset any time switches S1 and S2 are allowed to return to their neutral positions. When either switch is toggled up or down, the counters begin counting down. An MC14082 4-input NAND gate is driven by U5's Q1 output and outputs from the two D flip-flops to produce a slow (1-Hz) clock for 4 counts. Then, U5 is disabled by the final D flip-flop divider, and the 555's 4-Hz clock rate is selected by a 2-input NAND gate. This rate continues until the switch is released, thus disabling the down-counters. A timing diagram is shown in Figure 10, which summarizes operation of the dual-rate clock circuit.

### Brightness and Contrast Counters

The DRC output drives the brightness- and contrast-setting counters. As illustrated in Figure 8, the DRC signal is applied to two NOR gates. These gates trigger the appropriate paired counters, U11 - U12 or U13 - U14,

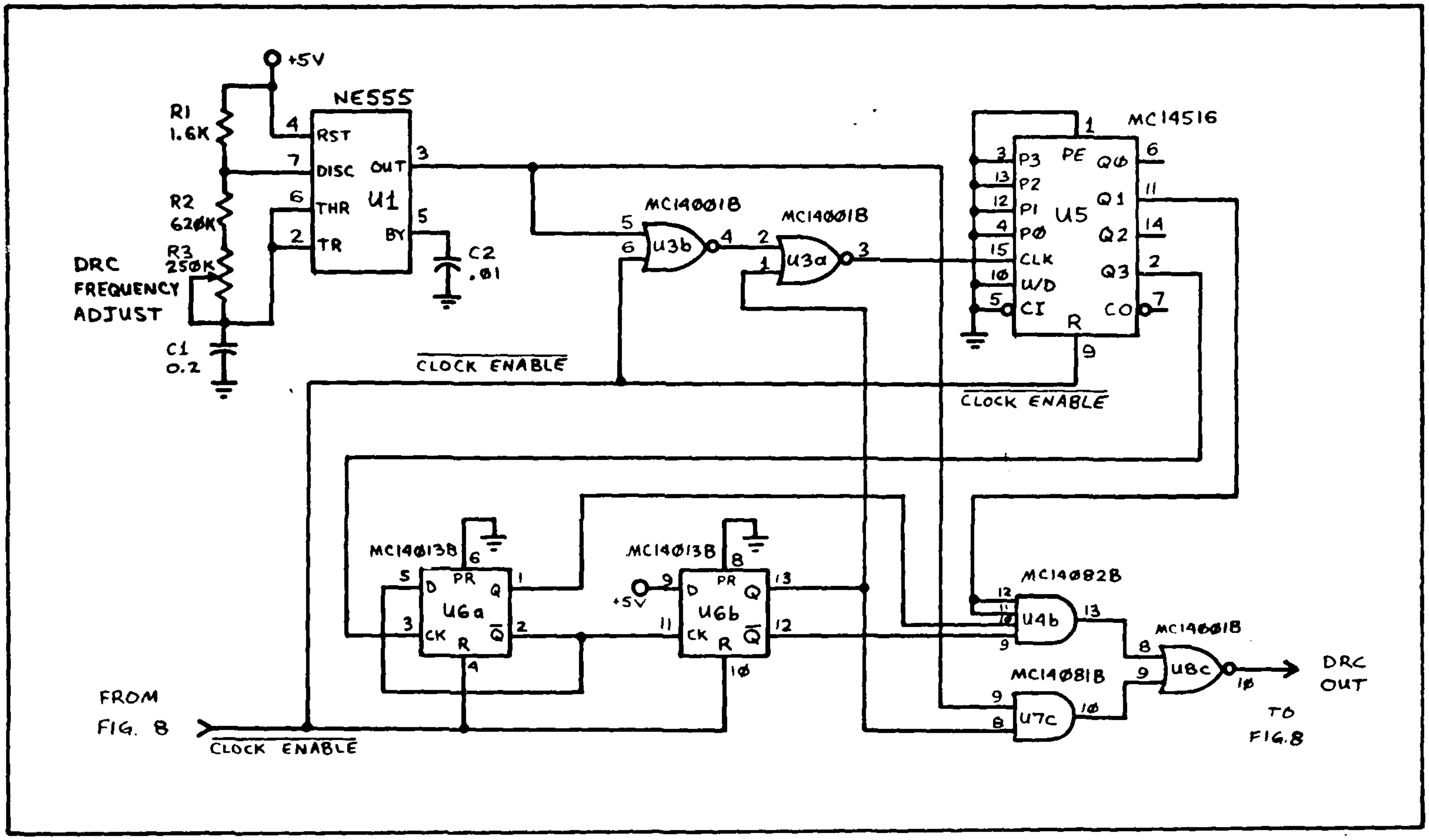


Fig 9. Dual-Rate Clock (DRC) Schematic

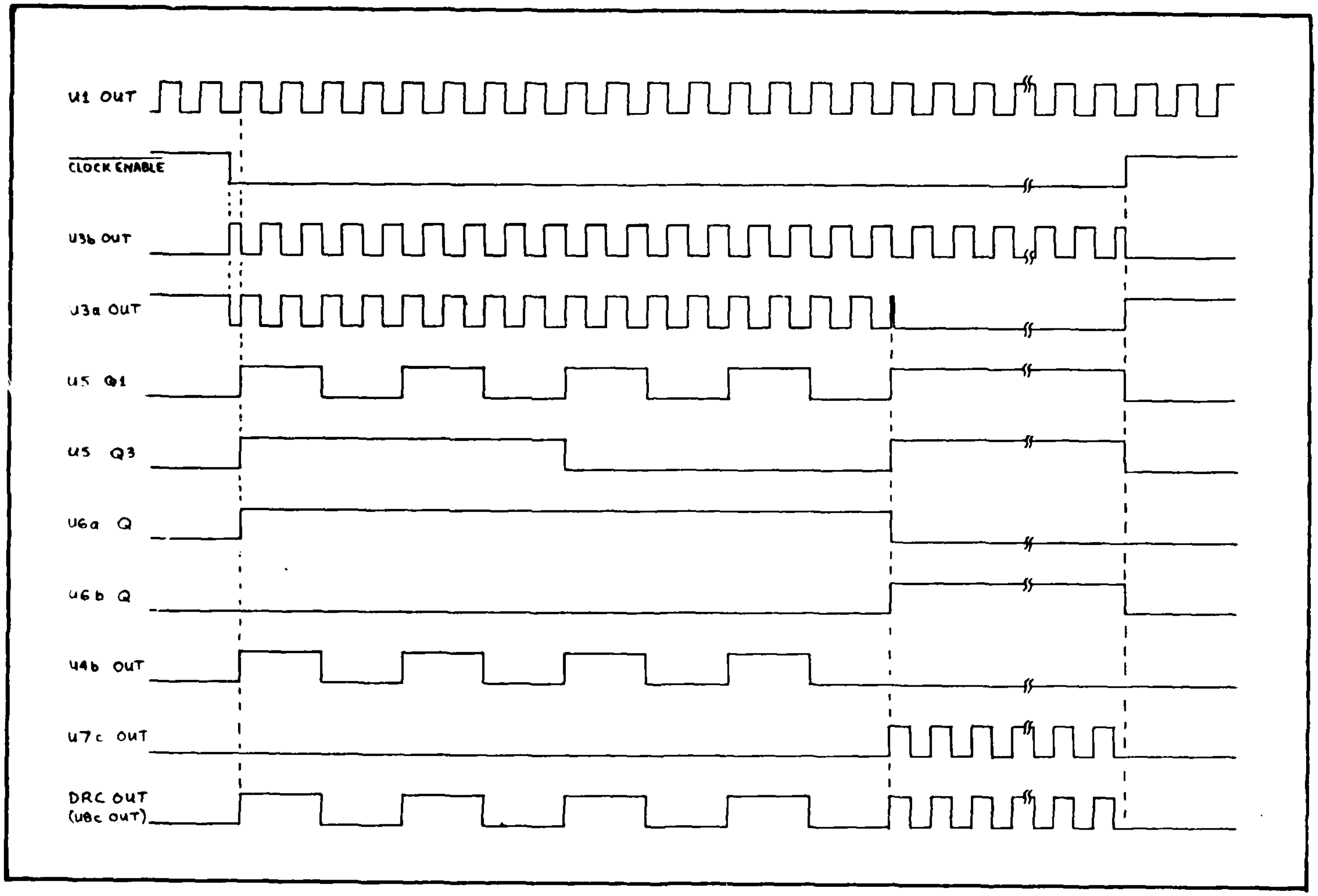


Fig 10. Dual-Rate Clock (DRC) Timing Diagram

depending on whether S1 or S2 is enabled. S1 selects brightness setting; S2 selects contrast setting.

Brightness is set in unit increments over the range of 0-39 foot-Lamberts (fL),while contrast is set in 1% increments over a range of 0-99%. Consequently, MC14510 BCD Up/ Down Counters were selected for counters U11, U13 and U14. An MC14516 Binary Up/ Down Counter was selected for U12 to simplify the hardware. In a binary counter, the least significant two bits, taken alone, count 0-1-2-3-0-1-... Thus by using only the Q0 and Q1 outputs of U12, it was possible to have the counter appear to reset after counting beyond 39. Similarly, when counting backwards from 00, the counter appears to preset to 39. Thus, brightness values can be reset to appropriate values when counting up or down, and additional hardware for resetting was unnecessary.

The counters are set in the count-up mode by applying a logical "1" value to the Up/Down-select pin of each counter. When a logical "0" signal is present at its Up/Down pin, a counter is switched to the count-down mode. By connecting the Up/Down-select pin of counters U11 and U12 to the BDN* output of switch debouncer, U2, both counters are set for counting up except when S1 is depressed (i.e. pushed down for counting down). Similarly, the Up/Down pins of U13 and U14 are connected to the CDN* output of U2, thus selecting them for count-up except when S2 is depressed.

The preset enable (PE) inputs for counters U11-U14 are connected to Output Status bit 2 (OS2) to allow preset values to be set at initialization. The preset values are user selectable via DIP switches S5 and S6. Note that the values should be set in binary-coded decimal (BCD) format. For example, suppose it is desirable to preset brightness to 32 fL, so that the system is automatically set to this value every time the system is brought up. Switch S5 should be set to 0-0-1-1-0-0-1-0. Similarly, desired contrast could be preset to 25% by setting switch S6 to 0-0-1-0-0-1-0-1.

Buffers U15, U16, and U17 provide the drive current required for interfacing the CMOS counters with the TTL bus interface circuits and for driving the TIL-311 displays.

### Flag Circuits

Flip-flops in U9 and U10, shown in Figure 8, generate flags to tell the user and computer the status of data entered. The two D-type flip-flops (FF) in U10 light up the left decimal point LED's in displays D1 and/or D3 when brightness and/or contrast settings, respectively, are changed. Once set, the LED's remain on until the computer has read the values and reset the flip-flops.

The flip-flops in U9 generate flags to tell the computer that brightness and/or contrast is or is not to be updated. When S3 is in the "down" position, FF1 of U9 remains "off", thus sending a logical "0" out on Input Status bit 0 (IS0). This tells the computer that its stored value for desired brightness (DBRT) is not to be updated. When S3 is toggled up, FF1 of U9 is set, sending a logical "1" to IS0. The computer recognizes this the next time it tests the input status bits. It updates DBRT and acknowledges this by toggling Output Status bit 0 (OS0) on and off. This clears the IS0 bit and the "brightness data changed" LED (left decimal point of display D1). If S3 is left in the "up" position, the computer updates DBRT and resets the two FF's on every pass.

Similarly, switch S4 signals when desired contrast (DCON) is to be updated. Toggling S4 up sets OS1. The computer detects this and resets OS1 and the "contrast data changed" LED (left decimal point of display D3) by toggling IS1 on and off.

## RETICON SAMPLING AND DIGITIZING CIRCUITS

There are actually several circuits. As outlined in Chapter III, their function is to produce digitized samples of the sine-wave gratings displayed on the video monitor. These circuits include a Scan Start Pulse Generator, the Reticon circuits, an Inverting Amplifier, an Analog-to-Digital Converter (ADC) circuit, an 8-bit latch, and a WAIT-State Controller (WSC). Figure 2 (in Chapter III) showed how these circuits interface with each other. Figure 11 is a timing diagram, showing the phase relationships of the various signals in these circuits.

### Scan Start Pulse Generator (SSPG)

This circuit is illustrated in Figure 12. When triggered by vertical synch pulses (from the video monitor) the SSPG generates Scan Start pulses for the Reticon circuits which are synchronized to the Reticon clock (RCLOCK). When triggered by the Scan Start pulse, the

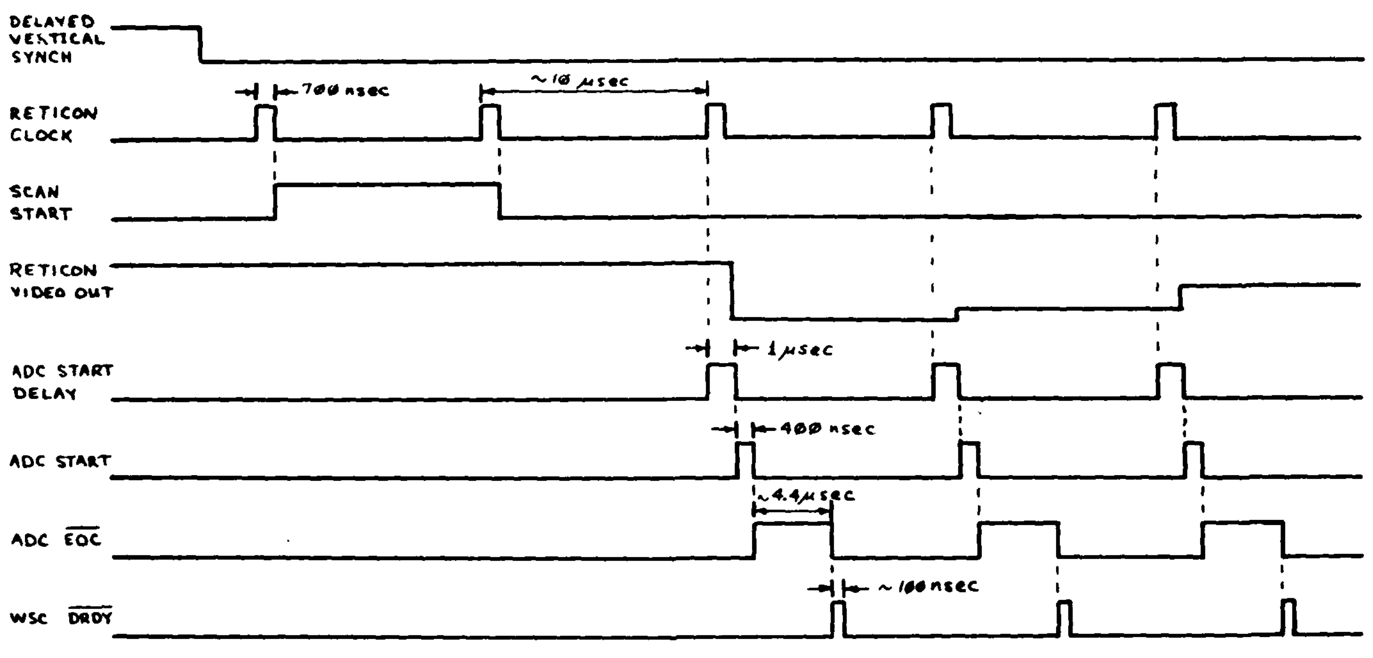


Fig 11. Reticon Circuits Timing Diagram

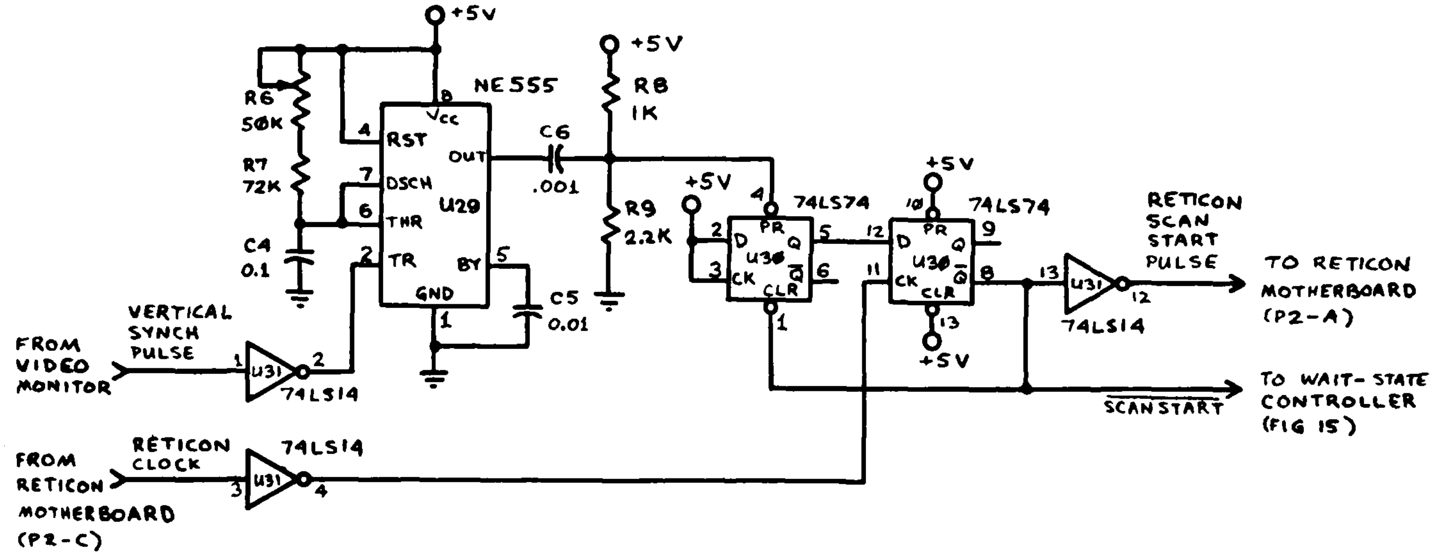


Fig 12. Scan-Start Pulse Generator (SSPG)

Reticon circuits scan the video monitor and produce a boxcar waveform on the Reticon motherboard's Video Out line.

U29 is a 555 timer, set up as a monostable multivibrator, or "one-shot". This device introduces a variable delay, once triggered by a vertical synch pulse from the video monitor. The delay is set by adjusting R7 so that a Reticon scan occurs in a "time-window", during which the video monitor's E-beam does not trace across the array. This delay timer is variable over a range of 8 to 13 msecs.

The rest of the SSPG is a "one-and-only-one" synchronizer circuit found in Lancaster's TTL Cookbook (Ref 9:209-210). This circuit produces a single start pulse, synchronized to the Reticon clock, each time it is triggered by a delayed vertical trigger pulse. Lancaster explains how this circuit works in his book.

ADC Start Pulse Generator

This circuit, depicted in Figure 13, synchronizes the ADC with the Reticon circuits. As shown in Figure 11, an ADC Start pulse is sent each time a new sample is ready to be digitized.

The first one-shot in Figure 13 produces a 1-usec pulse when triggered by the rising edge of a Reticon clock pulse. This is simply a delay to allow the new sample to become stable before being digitized by the ADC. The second one-shot generates a 400-nsec pulse, when triggered by the falling edge of the first one-shot's output. This is the start pulse, which triggers the ADC. The NOR gate, in

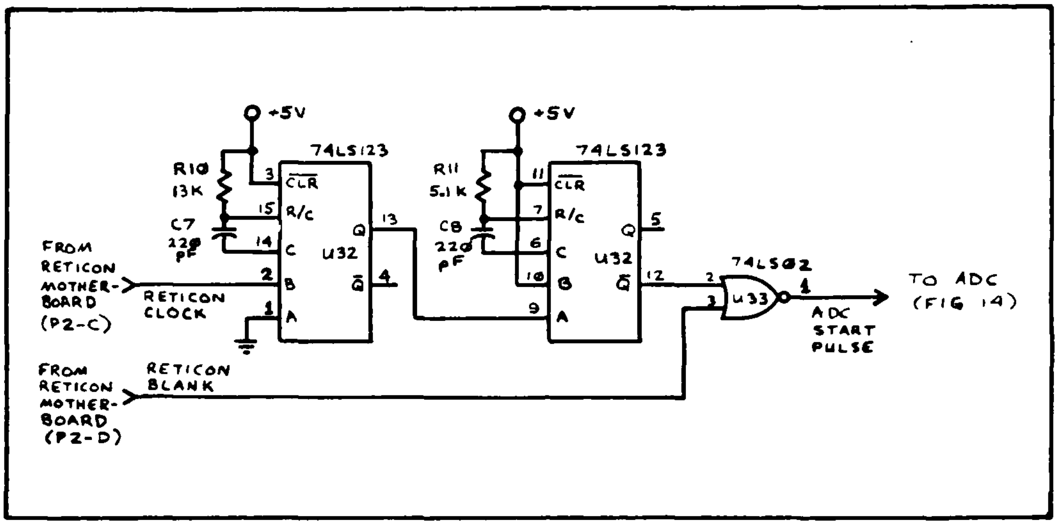


Figure 13. ADC Start Pulse Generator

Figure 13, is controlled by the Reticon's Blanking Pulse, which is low during scans, and high otherwise. Thus, ADC Start Pulses are generated only during scans of the monitor.

### Inverting Amplifier and Analog-to-Digital Converter

These circuits are shown in Figure 14. The Inverting Amplifier is simply a 741 op-amp configured as an inverter. R14 sets the gain of the inverting amplifier. R15 was selected to minimize the input bias current error (Ref 10:AN20-1).

U35 is a Datel ADC-HZ12BGC analog-to-digital converter, set for 8-bit short-cycled operation. This circuit was set up as outlined in Datel's Product Handbook (Ref 11:44-47).

U35 begins a conversion when triggered by the ADC Start Pulse Generator. The End-of-Conversion (EOC') output goes

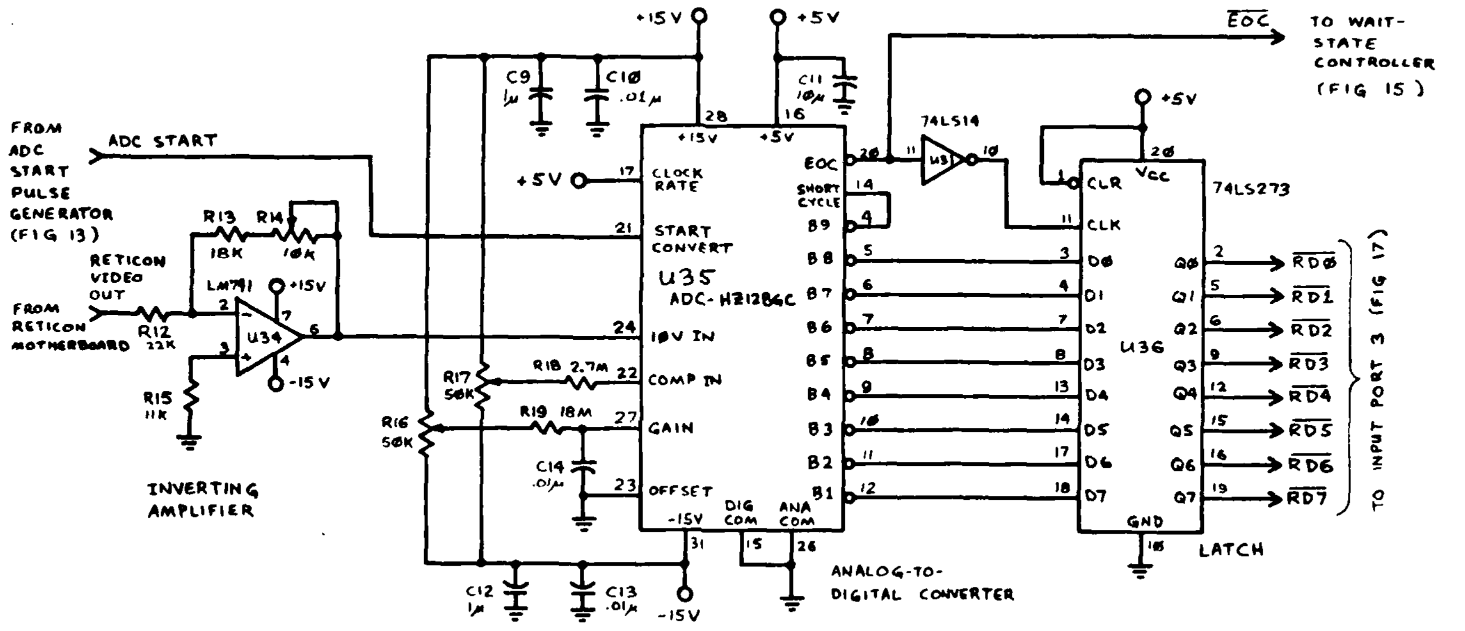


Fig 14. Inverting Amplifier and A/D Converter

low as soon as a conversion is completed, and valid data is present on the ADC's output. This transition triggers the 8-bit latch in Figure 14, which saves the digitized sample until the next sample is ready. The latch was used because invalid data are present on the ADC outputs during conversions. The latch is connected to input port 3, the Reticon Data Port (RDPORT). The EOC' signal is also sent to the WAIT-State Controller, described in the next section.

## WAIT-STATE CONTROLLER (WSC) CIRCUIT

This circuit synchronizes the Z80 processor with the Reticon scanning/digitizing circuits. This synchronization is necessary because the Z80 and the Reticon circuits run at different clock rates. Using the INIR instruction, the Z80 is able to read samples every 8.4 usecs. For timing purposes, the Reticon clock is then set so that each clock cycle is slightly longer than the Z80's "read cycle". For example, during testing, the Reticon clock was set for a 10-usec period. After reading each sample, the WSC then forced the Z80 to wait an additional 1.6 usecs until the next Reticon sample is ready.

In addition to synchronizing the processor with the sampling circuits during each sampling interval, the WSC also synchronizes the beginning of a Z80 "scan" (i.e. attempt to read a Reticon scan) with the start of the next Reticon array scan. The Reticon circuit begins a new monitor scan every 16.7 msecs (since it is triggered by the video monitor's vertical synch pulses), whereas the Z80 processor attempts to read a new scan only after performing a number of other tasks. Consequently, the processor only reads every third or fourth Reticon scan. If the Z80 attempts to read a scan after a scan has already begun, the WSC must force the processor to wait for the beginning of the next array scan before reading samples, even though the ADC circuit sends EOC pulses signalling that samples are ready to be read.

WSC Interaction With Other Circuits

Figure 15 contains a functional diagram of the WSC circuit, while Figure 16 contains a timing diagram which further illustrates its operation.

The WSC forces the Z80 into a "wait state" by placing a logical "0" on the processor's WAIT' line. When this occurs, the processor ceases all activity until the WAIT' line is brought high. When the WSC receives an EOC' pulse from the ADC circuit, it delays a moment to allow the data to be saved in the latch, then it places a logical "1" on the WAIT' line so that the Z80 can read the new value.

When the Z80 is ready to read data from the next array scan, it first toggles the WTBIT line (output status bit 3) off and on. This resets the WSC. When the Z80 then attempts it's first input (INIR) from the RDPORT, the WSC forces the processor into a wait mode until the following sequential events occur:

(1) the SSPG sends a Scan Start Pulse, indicating a new scan is to begin; and

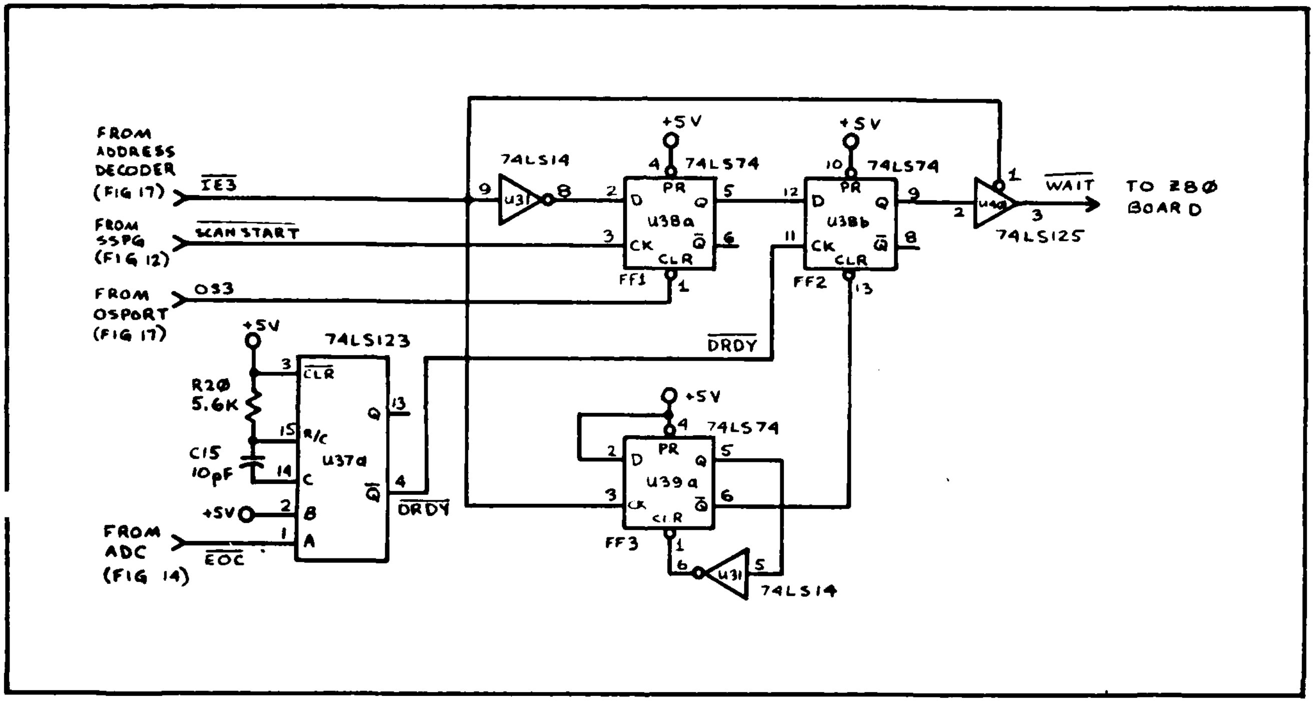
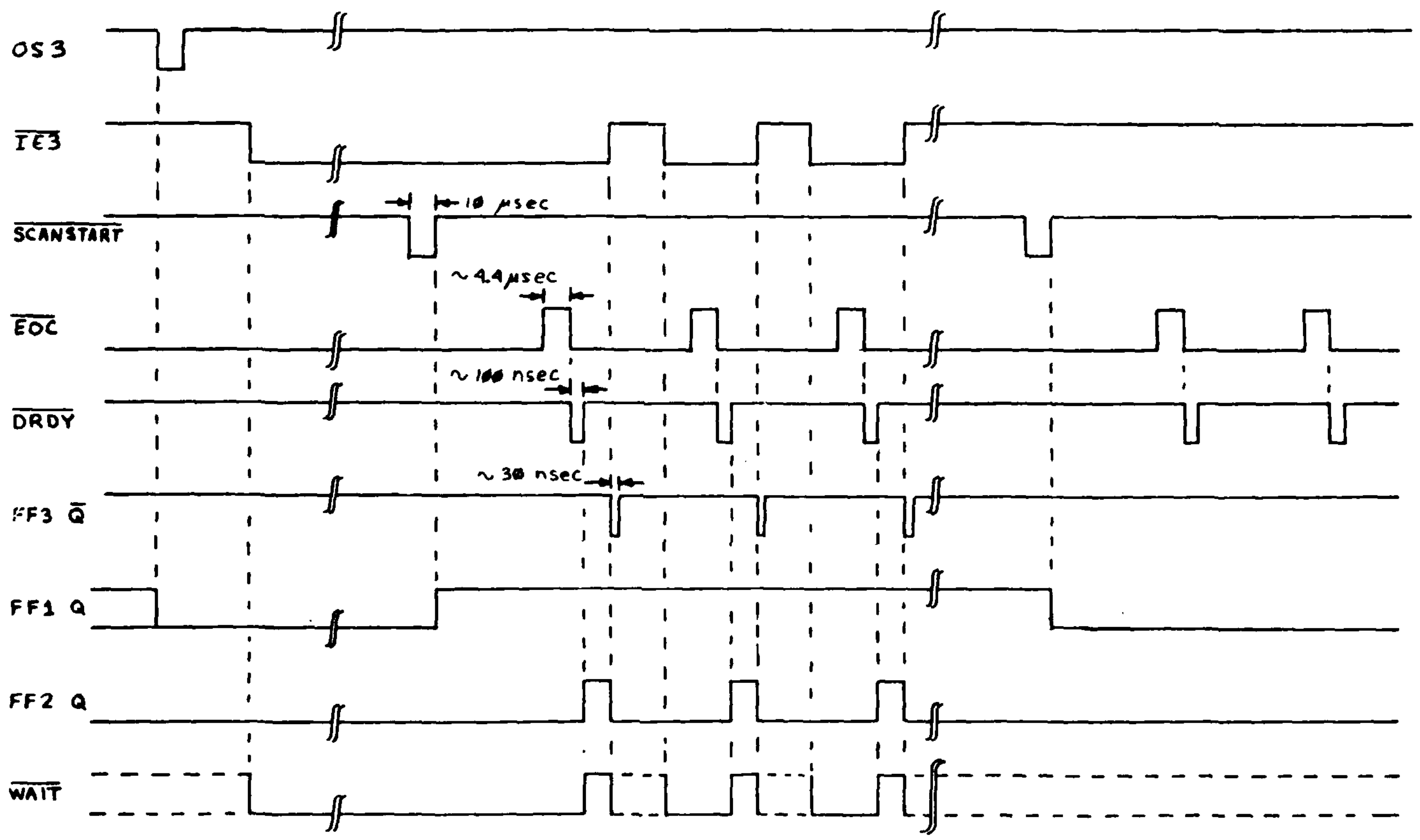


Fig 15. WAIT-State Controller (WSC) Schematic



NOTE: PULSE WIDTHS NOT TO SCALE

Fig 16. WAIT-State Controller (WSC) Timing Diagram

(2) the ADC circuit sends an EOC pulse, indicating the first sample has been digitized.

The WSC then places a logical "1" on the WAIT' line, and the processor begins reading data. After the first value has been read, the WSC simply inserts "wait states" as needed to synchronize the sampling circuits and processor for the rest of the scan.

## WAIT-State Controller Operation

When the computer is ready to read data from the next array scan, it first resets the wait state controller by toggling the WTBIT line (output status bit 3) off and on. This resets FF1 of Figure 15. The Z80 then executes an INIR command and attempts to read input port 3 (the RDPORT). As the Z80 attempts to read the RDPORT, a logical "0" is placed on the IE3' line. This enables output buffer U40, which places a logical "0" on the Z80's WAIT' line, forcing the processor to stop all activity until the WAIT' line goes high.

FF1 and FF2 both remain reset until the next Reticon scan start pulse is sent. This sets FF1, which remains set until cleared by the next WTBIT strobe or the next start pulse with IE3' high. With its D-line enabled, FF2 is set by the first low-to-high transition of DRDY', where DRDY' is simply a delayed EOC' pulse from the DAC circuit. The delay is provided to allow the Reticon sample to be latched and stable at input port 3. As the DRDY' line sets FF2, the Z80's WAIT' line is pulled high and the processor is allowed to read input port 3. After the port has been read, IE3' goes high, causing the WAIT' line to go to its high impedance state. In addition, this low-to-high transition of IE3' causes FF3 to reset FF2. It should be noted that FF3 is set up to send a short reset pulse to FF2 each time IE3' is active (low).

When the Z80 attempts to read the next sample at port 3, IE3' again goes low and enables the WAIT' line. The WAIT' line stays low until triggered by the next DRDY' pulse. The WAIT' line then goes high, the input instruction is executed, and the WAIT' line goes back to its high impedance state. This cycle repeats until the last data sample for the scan is read.

It should be noted that this sequence of events requires the sampling circuits to run slower than the Z80. If samples were ready faster than the Z80 could read them, the processor would begin skipping every other sample. After the Reticon circuit completed one scan of the array, the Z80 would still be looking for additional samples. It would then read the remaining samples from the next array scan. Consequently, some data would be lost, some would be counted twice, and overall accuracy would be lost.

This problem is eliminated by simply setting the Reticon's internal clock to have a period greater than the 8.4 microseconds required by the Z80 for each INIR read cycle. A clock period of 10 usecs worked well during testing.

## INPUT/OUTPUT (I/O) CIRCUITS

The design for the I/O circuits, shown in Figure 17, was derived from a book by Sargent and Shoemaker (Ref 12:48-53). Some minor modifications were made; but, these circuits are very similar to theirs.

Ports are selected by the address decoder, which consists of U18, a 74LS155 (Dual 2- to 4-Line Decoder), and several gates. The output of U8b becomes high only when WR' and IORQ' are both low (i.e. when the Z80 sends data to an output port). Similarly, the output of U8d becomes low only when RD' and IORQ' are both high (i.e. during a Z80 input). These output and input requests, however, only enable U18 when the output of U7a is high. This will occur only when A2 through A7 on the Z80 Address Bus are all low (i.e. when the Z80 addresses output or input ports 0 to 3). When enabled during a Z80 input or output attempt, U18 simply determines which of the input or output ports are addressed.

U19 is a 74LS245 Octal Bus Transceiver. This is nothing more than a bi-directional buffer. The direction data is transmitted is governed by U19's DIR line (pin 1). When a logical "1" is placed on this line, data is sent from U19's "A" side to its "B" side (i.e. from the Z80 side to the port side). A logical "1" on this line, sets the buffer for sending data from the "B" bus to the "A" bus. By connecting the DIR line to INREQ', the buffer is always set for output except during a valid input attempt (a port between 0 and 3 is addressed).

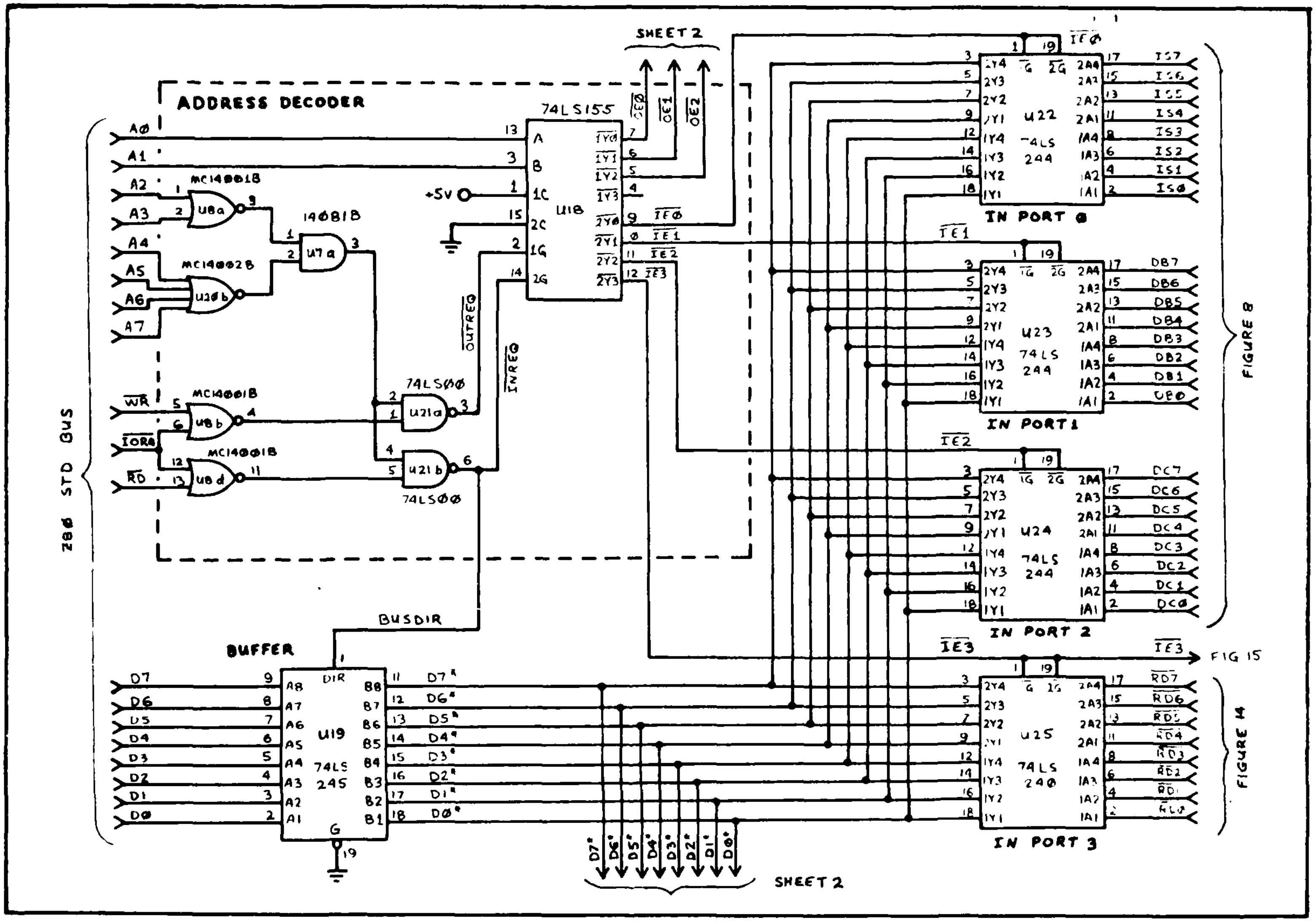


Fig 17. (Sheet 1) Input/Output (I/O) Circuit Schematic

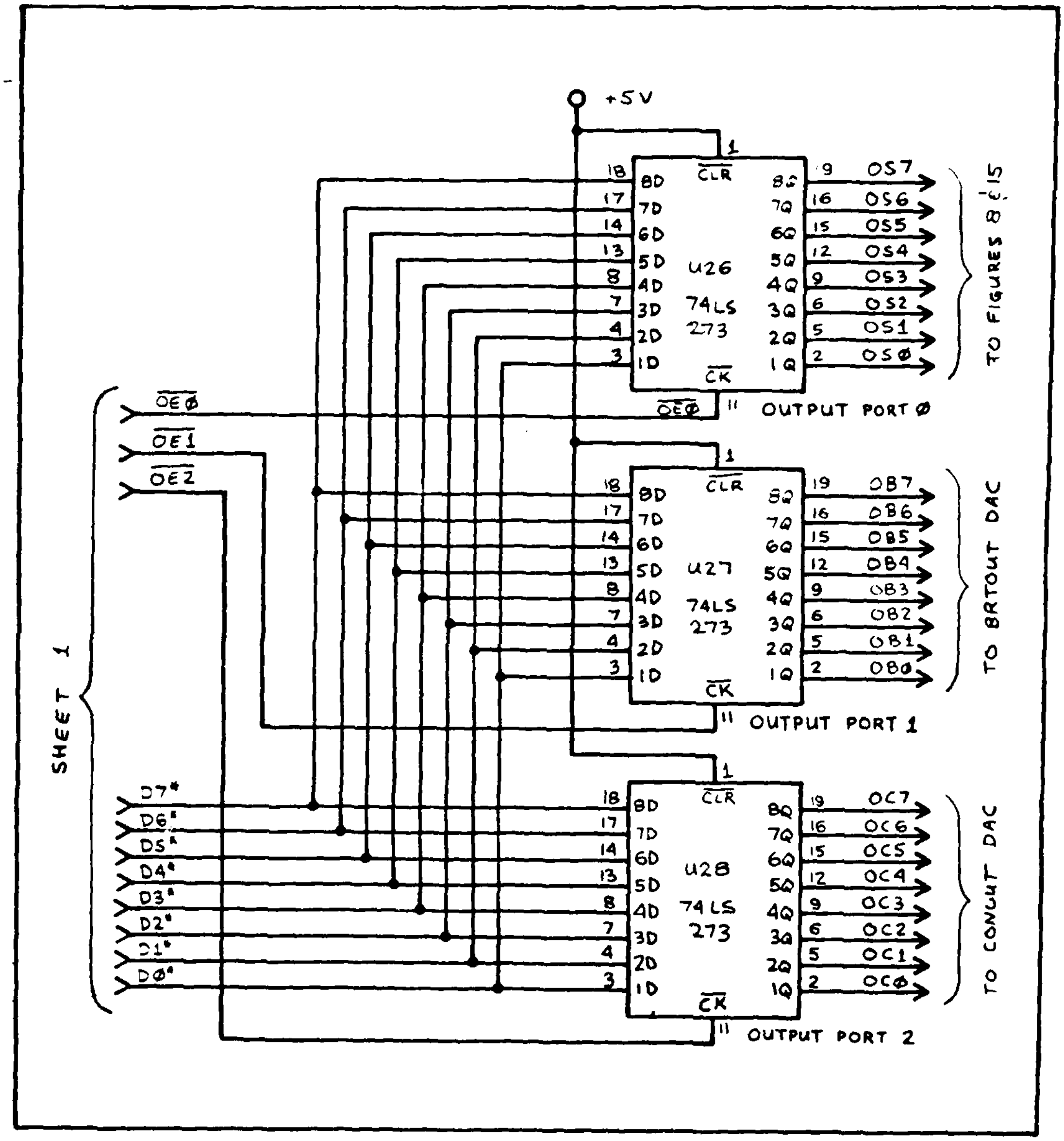


Fig 17. (Sheet 2) Input/Output (I/O) Circuit Schematic

U22, U23 and U24 (input ports 0, 1 and 2) are 74LS244 Octal Buffers. U25 (input port 3) is a 74LS240 Inverting Octal Buffer. These buffers are tri-state devices, thus they will only send data onto the data bus when their enable lines (1G' and 2G') are both low.

U25, U26 and U27 (output ports 0, 1 and 2) are 74LS273 Octal D Flip-Flops. A 74LS273 latches data from the buffered data bus (D0* through D7*) to its output lines when strobed by a rising edge on its clock line (pin 11). Once latched, the data byte remains on the output lines until a new byte is strobed into the latch.

To summarize operation of these circuits, suppose the Z80 attempts to send data to output port 0. To accomplish this, the Z80 places logical 0's on its address lines, A0 through A7. In addition, the Z80 places a data byte on its data bus. The WR' and IORQ' lines are then brought low as the Z80 attempts to write to port 0. U18 is enabled and sends a logical "0" out on OE0' (all U18 outputs are active low). The Z80 then brings its WR' and IORQ' lines high. This latches the data byte from the buffered data bus (the data bus buffer is set for output, since INREQ' is high at all times except during Z80 inputs) into output port 0. This data byte will then remain latched on bits OS0 to OS7 until the Z80 sends another byte to that port.

Now assume the Z80 attempts to read input port 0. As before, the address lines A0 to A7 are brought low. This time the Z80 brings its RD' and IORQ' lines low. U18 is again enabled, and a logical "0" is placed on IE0'. This enables the tri-state outputs of input port 0, placing a data byte onto the buffered data bus. The bus buffer was set to input when INREQ' went low, therefore the data byte is then read into the Z80. The Z80 then brings its RD' and IORQ' lines high. U18 becomes disabled and IEO' goes high. For detailed timing diagrams of Z80 input/output cycles, see Barden (Ref 6:35-36) or Saigent and Shoemaker (Ref 12:41-43).

## SUMMARY

This completes discussion of the hardware developed in this thesis. Table I is a summary list of parts used. Martindale and Lawson provided designs for the remaining circuits which must be constructed for the controller. These circuits include two Digital- to-Analog Converters, a Luminance Processor, and a Brightness Control Amplifier. Their designs will not be repeated here. It should be noted that Lawson constructed both an 8-bit and a 12-bit DAC. The 12-bit converter must be replaced with an 8-bit DAC. The Datel DAC-UP8BM used in Lawson's thesis appears to be a satisfactory device and would probably be acceptable for use in both converter DAC circuits.

TABLE I

PARTS LIST

| Schematic Reference | Device Type | Functional Description |
|---|---|---|
| U1 | NE555 | Timer IC |
| U2 | MC14490 | Hex Contact Bounce Eliminator |
| U3 | MC14001B | Quad 2-Input NOR Gate |
| U4 | MC14082B | Dual 4-Input AND Gate |
| U5 | MC14516B | Binary Up/Down Counter |
| U6 | MC14013B | Dual D Flip-Flop |
| U7 | MC14081B | Quad 2-Input AND Gate |
| U8 | MC14001B | Quad 2-Input NOR Gate |
| U9 | MC14013B | Dual D Flip-Flop |
| U10 | MC14013B | Dual D Flip-Flop |
| U11 | MC14516B | Binary Up/Down Counter |
| U12 | MC14510B | BCD Up/Down Counter |
| U13 | MC14510B | BCD Up/Down Counter |
| U14 | MC14510B | BCD Up/Down Counter |
| U15 | MC14050B | Hex Buffer |
| U16 | MC14050B | Hex Buffer |
| U17 | MC14050B | Hex Buffer |
| U18 | 74LS155 | Dual 2- to 4-Line Decoder |
| U19 | 74LS245 | Octal Bus Transceiver |
| U20 | MC14002B | Dual 4-Input NOR Gate |
| U21 | 74LS00 | Quad 2-Input NAND Gate |
| U22 | 74LS244 | Octal Tri-State Buffer |
| U23 | 74LS244 | Octal Tri-State Buffer |
| U24 | 74LS244 | Octal Tri-State Buffer |
| U25 | 74LS240 | Octal Inverting Tri-State Buffer |
| U26 | 74LS273 | Octal D Flip-Flop |
| U27 | 74LS273 | Octal D Flip-Flop |
| U28 | 74LS273 | Octal D Flip-Flop |
| U29 | NE555 | Timer IC |
| U30 | 74LS74 | Dual D Flip-Flop |
| U31 | 74LS14 | Hex Schmitt-Trigger Inverter |
| U32 | 74LS123 | Dual Monostable Multivibrator |
| U33 | 74LS02 | Quad 2-Input NOR Gate |
| U34 | LM741 | Op Amp |
| U35 | ADC-HZ12BGC | 12-Bit A/D Converter |
| U36 | 74LS273 | Octal D Flip-Flop |
| U37 | 74LS123 | Dual Monostable Multivibrator |
| U38 | 74LS74 | Dual D Flip-Flop |
| U39 | 74LS74 | Dual D Flip-Flop |
| U40 | 74LS125 | Quad Tri-State Bus Buffer |

TABLE I (cont.)

PARTS LIST

| Schematic Reference | Device Type | Functional Description |
|---|---|---|
| R1 | 1.6K | Resistor |
| R2 | 250K | Potentiometer |
| R3 | 620K | Resistor |
| R4 | 898-3-R1K | Resistor Array, 1K |
| R5 | 898-3-R1K | Resistor Array, 1K |
| R6 | 50K | Potentiometer |
| R7 | 72K | Resistor |
| R8 | 1K | Resistor |
| R9 | 2.2K | Resistor |
| R10 | 13K | Resistor |
| R11 | 5.1K | Resistor |
| R12 | 22K | Resistor |
| R13 | 18K | Resistor |
| R14 | 10K | Potentiometer |
| R15 | 11K | Resistor |
| R16 | 50K | Potentiometer, Cermet |
| R17 | 50K | Potentiometer, Cermet |
| R18 | 2.7M | Resistor |
| R19 | 18M | Resistor |
| R20 | 5.6K | Resistor |
| C1 | .2 uF | Capacitor, Ceramic |
| C2 | .01 uF | Capacitor, Ceramic |
| C3 | .001 uF | Capacitor, Ceramic |
| C4 | .1 uF | Capacitor, Ceramic |
| C5 | .01 uF | Capacitor, Ceramic |
| C6 | .001 uF | Capacitor, Ceramic |
| C7 | 220 pF | Capacitor, Ceramic |
| C8 | 220 pF | Capacitor, Ceramic |
| C9 | 1 uF | Capacitor, Electrolytic |
| C10 | .01 uF | Capacitor, Ceramic |
| C11 | 10 uF | Capacitor, Electrolytic |
| C12 | 1 uF | Capacitor, Electrolytic |
| C13 | .01 uF | Capacitor, Ceramic |
| C14 | .01 uF | Capacitor, Ceramic |
| C15 | 10 pF | Capacitor, Ceramic |

# APPENDIX B

## SOFTWARE DESCRIPTION

This appendix contains a description of the software routines listed in Appendix D. In addition, reference tables are provided at the end of this appendix. Table II lists the ports used by the Z80. Table III summarizes the various bits status bits used used in the ISPORT and OSPORT. Finally, Table IV lists the storage locations for variables used in the Z80 control software. To further assist the reader in understanding the algorithms used in these routines, flowcharts are included in Appendix C. A data dictionary is provided in Appendex E.

### INITIALIZATION ROUTINE (INIT)

This routine initializes the controller when the system is first powered up. First, INIT initializes the stack pointer to FB00H. Next, three pages of RAM are cleared to provide a "clean" scratchpad work area. The three pages include one page of storage for variables and two pages (512 bytes) of storage for samples read from a scan of the Reticon array.

Once the Z80 has been prepared, INIT initializes the external hardware by toggling all output status bits (except bit 3, which is used to signal that the Z80 is ready to read a Reticon scan) on and off. This clears any flags that might be arbitrarily set when the system is powered up. In addition, this presets the desired brightness and contrast counters to the values specified by switches S5 and S6.

Finally, INIT calls the BUPDAT and CUPDAT routines. This initializes the Z80's internal values for desired brightness and contrast (DBRT and DCON) to the preset values.

CLEAR MEMORY ROUTINE (CLRMEM)

This routine is called by INIT to clear a block of memory. CLRMEM was taken directly from Wadsworth (Ref 13:90). When called, CLRMEM assumes the starting address of the block to be cleared is in the HL register pair. In addition, register pair BC is assumed to contain the number of bytes to be cleared. It should be noted that CLRMEM affects the HL and BC register pairs. When exiting this routine, BC is cleared and HL contains the address of the first byte following the block that was cleared.

EXECUTIVE ROUTINE (EXEC)

This is the main program. Once entered, EXEC repeats in an endless loop. First, STEST is called to update desired brightness and contrast values, if the user signals that an update is desired. Next, SCAN is called to read all values from the next scan of the Reticon array. The samples read by SCAN are stored in a 512-byte data table. AVGBRT and CNTRST are then called to compute actual brightness and contrast from the Reticon data. Finally, BADJ and CADJ compare computed values for brightness and contrast with desired values. Corrections are sent to the brightness and contrast output circuits if computed and actual values disagree. EXEC then calls STEST again as the loop repeats itself.

STATUS UPDATE ROUTINE (STEST)

This routine allows the user to update the desired values for contrast and brightness. When called, STEST checks the input status port (input port 0) to see if the user wishes to update brightness and/or contrast. Bit 0 is first tested to see if brightness is to be updated. If so, BUPDAT is called. Next, bit 1 is tested to see if contrast is to be updated. If this bit is set, CUPDAT is called. The routine then returns control to the executive.

BRIGHTNESS UPDATE ROUTINE (BUPDAT)

This routine updates the stored value for desired brightness, DBRT. First, the input brightness port (input port 1) is read. The value read is saved as the new unscaled value for desired brightness, UDBRT. As this is a binary-coded decimal value, CONVRT is called to convert it to its binary equivalent. The binary form is then scaled for direct comparison with the computed value for actual brightness. Finally, the routine signals the hardware that desired brightness has been updated by toggling bit 0 of the output status port (output port 0) on and off.

The binary form of UDBRT is scaled by multiplying it by 256 / 40. This scale factor was chosen to provide maximum sensitivity to desired values. For example, a maximum brightness of 40 fL would correspond to a scaled binary value of 100 Hex. Actually, brightness is limited to

39 fL, therefore the maximum scaled value for desired brightness, DBRT, is 39 X 256 / 40 = 250 (or FA Hex).

## CONTRAST UPDATE ROUTINE (CUPDAT)

This routine is very similar to BUPDAT. When called, CUPDAT reads the input contrast port (input port 2). The value read is saved as the new unscaled value for desired contrast, UDCON. CONVRT is called to transform UDCON to its binary equivalent. The binary form is then scaled and the result is saved as the updated value for desired contrast, DCON. Finally, the routine signals the hardware that desired contrast has been updated by toggling bit 1 of the output status port (output port 0) on and off.

The binary form of UDCON is scaled by multiplying it by 256 / 100. Again, the scale factor was chosen to provide maximum sensitivity to desired values. Ideally, a maximum contrast value of 100% would correspond to a scaled binary value of 100 Hex. Actually, desired contrast is limited to 99%, therefore the maximum scaled value for DCON is 99 X 256 / 100 = 253 (or FD Hex).

## DECIMAL TO BINARY CONVERSION ROUTINE (CONVRT)

This routine converts the 8-bit binary-coded decimal number in register A to its binary equivalent. As shown in Appendix C, CONVRT simply adds the least significant digit (digit2) to ten times the most significant digit (digit1). The result is left in register A.

## DIVIDE ROUTINE (DIVIDE)

This routine divides a 16-bit unsigned integer dividend in the HL register pair by an 8-bit unsigned integer divisor in register C. The 8-bit quotient is left in register L. It should be noted that this routine rounds the least significant bit of the quotient up one bit if the remainder (in register H) is greater than half the divisor. This feature was added to avoid loss of precision which results when large remainders are simply truncated.

Several 16/8-bit division routines (Ref 6:190, 7:8-12, 8:135) were tried before this routine was developed. Unfortunately, each of these routines failed whenever a dividend larger than 7F Hex was used (i.e. whenever bit 15 was set). The problem with all of the division routines tested is that when the dividend (and divisor) is shifted left, a logical "1" in bit 15 is shifted into the carry bit. This bit is "lost" in the subtraction which follows the shift left operation. This problem can be overcome by testing for overflow each time the dividend (and quotient) is shifted left. If an overflow is encountered when shifting the dividend left, the dividend must be larger than the divisor, so the divisor is subtracted automatically.

The basic algorithm for DIVIDE was borrowed from Leventhal (Ref 7:8-12). Leventhal's algorithm was modified as described above by testing for overflow each time HL is shifted left. The flowchart in Appendix C illustrates the overall algorithm used in DIVIDE.

## SCAN ROUTINE

This routine reads the data samples from a single scan of the Reticon array. The routine was developed to read quickly, without "handshakes" back and forth to acknowledge receipt of data. By using the block input with increment command, INIR, it was possible to read each value in as little as 8.4 microseconds.

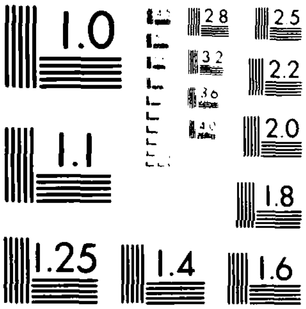
SCAN begins by sending a reset pulse to the wait state controller (Bit 3 of output status port is toggled off and on). This helps synchronize the external hardware with the Z80's attempts to read the array scan data. Next, registers B, C, H, and L are initialized for the INIR command. Register B is cleared to indicate a block of 256 bytes are to be read. Register C is set to the proper input port. The HL register pair is aimed at TABLE, the starting address of the block in RAM that is to hold the Reticon samples. The INIR command then reads the first 256 bytes of data and stores them in the first page of the TABLE block of memory. After the first page of data is read, the routine immediately reads the remaining 256 samples, then returns control to the executive routine.

## BRIGHTNESS CALCULATION ROUTINE (AVGBRT)

This routine computes the average brightness, ABRT, of the samples stored in the Reticon-scan data table, TABLE. Initially, SUMTBL is called to determine the overall sum of the 512 samples stored in TABLE. The table sum, TSUM, is then divided by 512 to compute the net average brightness,

AD-A135 871    A SINE-WAVE GRATING CONTROLLER FOR VISION TESTING(U)    2/2
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL
OF ENGINEERING  B D BAXLEY MAR 83 AFIT/GE/EE/83M-1

UNCLASSIFIED    F/G 14/2    NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ABRT. To speed computation time, the division is performed by effectively shifting the 17-bit TSUM right nine places. The 8-bit result is rounded up one bit if the remainder is greater than or equal to 256. The final result is saved as ABRT.

TABLE SUMMING ROUTINE (SUMTBL)

This routines sums the 512 samples stored in TABLE. As speed of execution was important in all routines, it was determined that the table could be summed quickest if added one page at a time. Therefore, a page summing subroutine, ADPAGE, was developed to add each of the two 256-byte pages of data separately, and SUMTBL became a routine which simply added the two page sums.

The routine initially aims register pair DE at the first byte of the first page to be summed. Then ADPAGE is called to add all bytes on that page. The sum of samples from the first page is returned by ADPAGE in the HL register pair. This sum is saved in the BC register pair and register pair DE is aimed at the beginning of the second page to be summed. ADPAGE is again called, and the second page sum is added to the first.

It was noted that the sum of each memory page could be stored in 16 bits, since the maximum sum for a page is 256 X 255 = 65,280 (or FF00 Hex). However, when the two page sums are added together, the maximum sum could exceed 16 bits (maximum sum for two pages is 1FE00 Hex). Thus, the table sum, TSUM, requires three bytes of storage, with the most significant byte containing only one bit of TSUM.

PAGE ADDITION ROUTINE (ADPAGE)

This routine sums all bytes of a memory page, starting with the byte addressed by the DE register pair and ending with the last byte of that page. Therefore, to add all bytes on a page, DE must address the first byte of the page to be summed.

Initially, the routine clears the BC and HL register pairs, as these will be used for addition. The routine then enters a loop whereby the byte addressed by DE is added to the sum in HL, register E is incremented to point to the next value, and a test is made to see if the last value on the current page has been summed. The loop repeats until register E is zeroed, (i.e. until DE has reset to the beginning of the page). The HL register pair then holds the net sum of all values on the page of interest. All other registers are restored and control returns to the calling program.

ACTUAL CONTRAST ROUTINE (CNTRST)

This routine computes the actual contrast of the grating on the monitor by solving

$$ACON = \frac{256 \text{ X } (BMAX - BMIN)}{(2 \text{ X } BAVG)} \tag{4}$$

The algorithm for this routine is shown in Appendix C.

First FMAX and FMIN are called to search the Reticon data and find the maximum and minimum brightness values, BMAX and BMIN. Then, these values and the average brightness, BAVG (= ABRT computed by AVGBRT), are used to compute ACON as in Eq (4) above. Actually, to speed up the computation, the numerator of Eq (4) is computed by shifting (BMAX - BMIN) left eight bits. DIVIDE then divides the numerator by BAVG, and the result is shifted right one bit to divide by two. The final result is saved as the actual contrast, ACON.

It should be noted that true contrast would be a percentage, and thus a fraction. To overcome this problem, ACON is actually scaled by pre-multiplying true contrast by 256 so that only integer values need be computed.

FIND MAX ROUTINE (FMAX)

This routine finds the maximum sample value in the Reticon data table. To speed computing time, the values are searched one page at a time, as shown in Appendix C. Other algorithms are possible, but they would add at least five clock cycles per comparison (e.g. using the CPI instruction with register pair BC serving as a counter). This would add about 1 msec overall, since there are over 500 comparisons.

It should be noted that the first two samples are ignored by both FMAX and FMIN. This was necessary because the first two samples produced by the Reticon are invariably of larger magnitude than the remaining samples (see discussion of this phenomenon in Chapter 6.

## FIND MIN ROUTINE (FMIN)

This routine is virtually identical to FMAX except that the routine searches for a minimum value instead of a maximum. As in FMAX, the Reticon data is searched one page at a time, starting with "TABLE + 2" (i.e. the first two samples are ignored). The algorithm for FMIN is depicted in Appendix C.

## BRIGHTNESS ADJUST ROUTINE (BADJ)

This routine compares values for actual brightness, ABRT, and desired brightness, DBRT, and sends a brightness correction to the brightness output port (output port 1) if ABRT and DBRT differ. This routine is only a prototype since the actual brightness correction circuits have not been built and tested.

BADJ begins by comparing ABRT and DBRT. If the two values match, the routine simply returns to the executive routine. If ABRT is greater than DBRT, the old value for brightness out, BRTOUT, is decremented by one and the new value for BRTOUT is sent to the OBPORT. If neither of the above conditions is met, ABRT is assumed smaller than DBRT. BRTOUT is incremented and the new BRTOUT is sent to the OBPORT.

This prototype routine is slow but should eventually converge to a steady-state value for actual brightness, whereby ABRT and DBRT match. Since BRTOUT is incremented only one bit each time it's called, the worst-case time for it to reach it's steady-state value is 255 X TEXEC, where

TEXEC = worst-case overall execution time of a single pass of the executive routine). From experimentation, it appears that TEXEC is approximately 40 msec. Therefore, BADJ should converge to a steady-state value for BRTOUT in about 10 seconds.

Faster correction algorithms are possible. For example, half the difference between ABRT and DBRT could be added to BRTOUT to speed convergence. However, the hardware has not yet been completed, therefore, tradeoffs of the various alternatives are not practical at this time. The current algorithm was selected for its simplicity and stability.

CONTRAST ADJUST ROUTINE (CADJ)

This routine is basically similar to BADJ. Values for desired and actual contrast are compared. If the two values agree, no correction is made to the contrast out, CONOUT, value sent to the contrast output port (output port 2). If ACON is greater than DCON, CONOUT is decremented by one. If ACON is less than DCON, CONOUT is incremented by one. As in BADJ, this is a prototype routine, selected for its simplicity and stability.

## TABLE II

## Z80 INPUT AND OUTPUT PORTS

### INPUT PORTS

| PORT | NAME | DESCRIPTION |
|---|---|---|
| 0 | ISPORT | INPUT STATUS PORT<br>- PORT USED TO INPUT STATUS FLAGS FROM CONTROLLER HARDWARE |
| 1 | IBPORT | INPUT BRIGHTNESS PORT<br>- PORT USED TO INPUT UDBRT VALUES FROM USER INTERFACE CIRCUITS |
| 2 | ICPORT | INPUT CONTRAST PORT<br>- PORT USED TO INPUT UDCON VALUES FROM USER INTERFACE CIRCUITS |
| 3 | IDPORT | INPUT DATA PORT<br>- PORT USED TO INPUT RETICON DATA FROM ADC CIRCUIT |

### OUTPUT PORTS

| PORT | NAME | DESCRIPTION |
|---|---|---|
| 0 | OSPORT | OUTPUT STATUS PORT<br>- PORT USED TO SEND OUTPUT CONTROL SIGNALS TO CONTROLLER HARDWARE |
| 1 | OBPORT | OUTPUT BRIGHTNESS PORT<br>- PORT USED TO SEND BRTOUT VALUES TO BRIGHTNESS DAC |
| 2 | OCPORT | OUTPUT CONTRAST PORT<br>- PORT USED TO SEND CONOUT VALUES TO CONTRAST DAC |
| 3 | - | NOT USED |

## TABLE III

## STATUS BIT USAGE

### INPUT PORT

| BIT | NAME | DESCRIPTION |
|---|---|---|
| 0 | BUFLG (IS0) | BRIGHTNESS UPDATE FLAG<br>- THIS FLAG SIGNALS THE COMPUTER THAT THE USER WISHES TO UPDATE THE STORED VALUE FOR DESIRED BRIGHTNESS |
| 1 | CUFLG (IS1) | CONTRAST UPDATE FLAG<br>- THIS FLAG SIGNALS THE COMPUTER THAT THE USER WISHES TO UPDATE THE STORED VALUE FOR DESIRED CONTRAST |
| 2-7 | - | UNUSED |

### OUTPUT PORT

| BIT | NAME | DESCRIPTION |
|---|---|---|
| 0 | BFCLR (OS0) | BRIGHTNESS FLAG CLEAR BIT<br>- THIS BIT RESETS THE BRIGHTNESS FLAGS (BUFLG AND BNUFLG) AFTER DBRT HAS BEEN UPDATED |
| 1 | CFCLR (OS1) | CONTRAST FLAG CLEAR BIT<br>- THIS BIT RESETS THE CONTRAST FLAGS (CUFLG AND CNUFLG) AFTER DCON HAS BEEN UPDATED |
| 2 | PRESET (OS2) | PRESET BIT<br>- THIS BIT INITIALIZES COUNTERS TO DEFAULT VALUES FOR DBRT & DCON WHEN THE SYSTEM IS RESET |
| 3 | RSCAN (OS3)<br>(ALSO WTBIT) | RETICON SCAN BIT<br>- THIS BIT RESETS THE WAIT-STATE CONTROLLER WHEN THE Z80 IS READY TO READ A RETICON SCAN |
| 4-7 | - | UNUSED |

## TABLE IV

## VARIABLE STORAGE LOCATIONS

| LOC'N | VALUE | DESCRIPTION |
|---|---|---|
| F800H | UDBRT | 2-DIGIT UNSCALED DESIRED BRIGHTNESS |
| F801H | DBRT | 8-BIT SCALED VALUE FOR DESIRED BRIGHTNESS |
| F802H | ABRT | COMPUTED ACTUAL BRIGHTNESS |
| F803H | BRTOUT | BRTNESS-OUT SETTING SENT TO OBPORT |
| F810H | UDCON | 2-DIGIT UNSCALED DESIRED CONTRAST |
| F811H | DCON | 8-BIT SCALED VALUE FOR DESIRED CONTRAST |
| F812H | ACON | COMPUTED ACTUAL CONTRAST |
| F813H | CONOUT | CONTRAST-OUT SETTING SENT TO OCPORT |
| F820H | OSTAT | LOCATION OF OUTPUT STATUS BYTE |
| F821H | ISTAT | LOCATION OF INPUT STATUS BYTE |
| F830H | BMAX | MAXIMUM SAMPLE VALUE |
| F831H | BMIN | MINIMUM SAMPLE VALUE |
| F832H | TSUM | TABLE SUM STORAGE AREA (3 BYTES) |
| F900H | TABLE | BEGINNING OF 512-BYTE RETICON DATA TABLE |

# APPENDIX C

## FLOWCHARTS

This appendix contains the flowcharts, which describe the software routines contained in Appendix D. A discussion of the algorithms for these routines is contained in Appendix B.

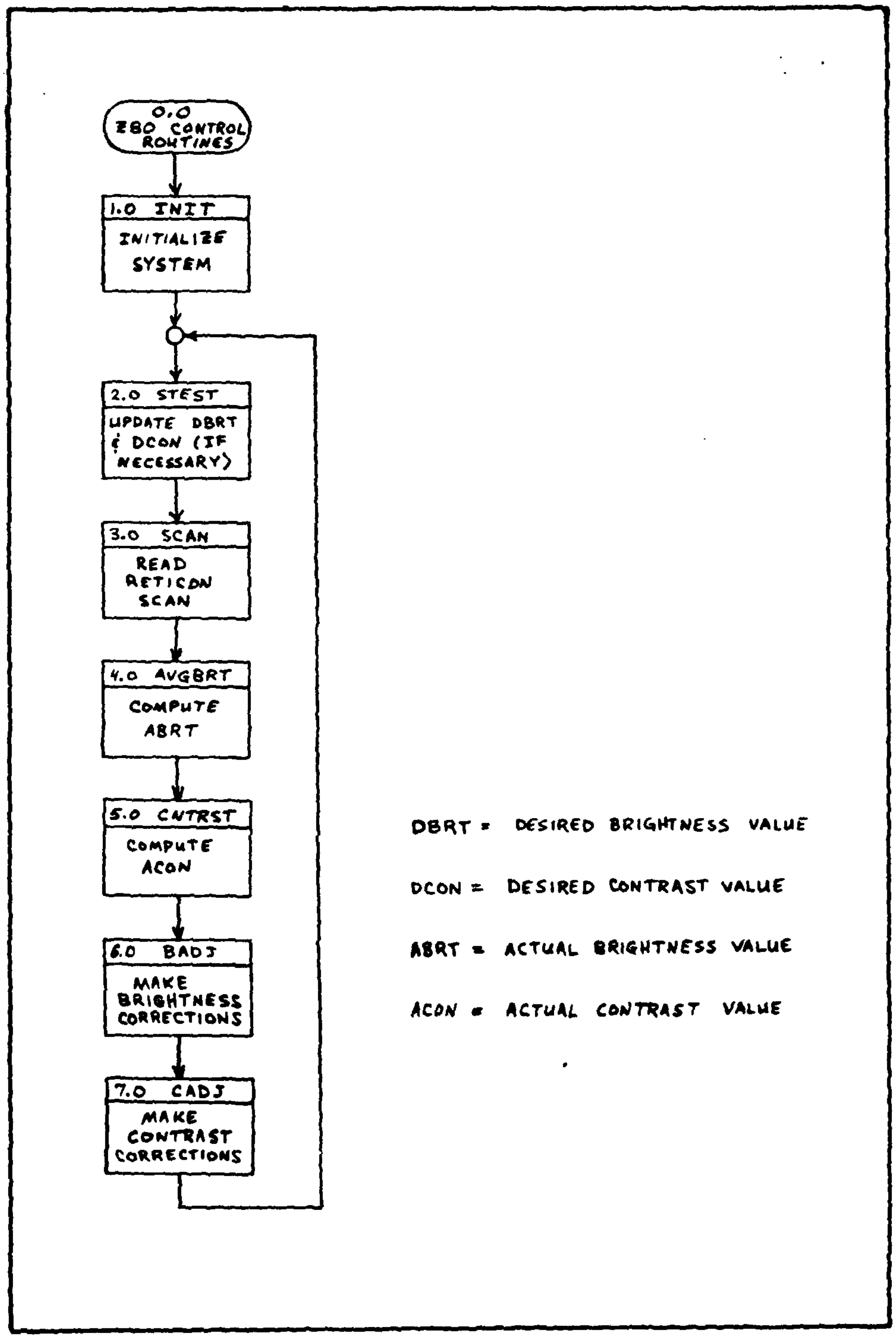

0.0 Z80 CONTROL ROUTINES
1.0 INIT
INITIALIZE SYSTEM
2.0 STEST
UPDATE DBRT & DCON (IF NECESSARY)
3.0 SCAN
READ RETICON SCAN
4.0 AVGBRT
COMPUTE ABRT
5.0 CNTRST
COMPUTE ACON
6.0 BADJ
MAKE BRIGHTNESS CORRECTIONS
7.0 CADJ
MAKE CONTRAST CORRECTIONS
DBRT = DESIRED BRIGHTNESS VALUE
DCON = DESIRED CONTRAST VALUE
ABRT = ACTUAL BRIGHTNESS VALUE
ACON = ACTUAL CONTRAST VALUE

Fig 18. Z80 Control Routine Flowchart

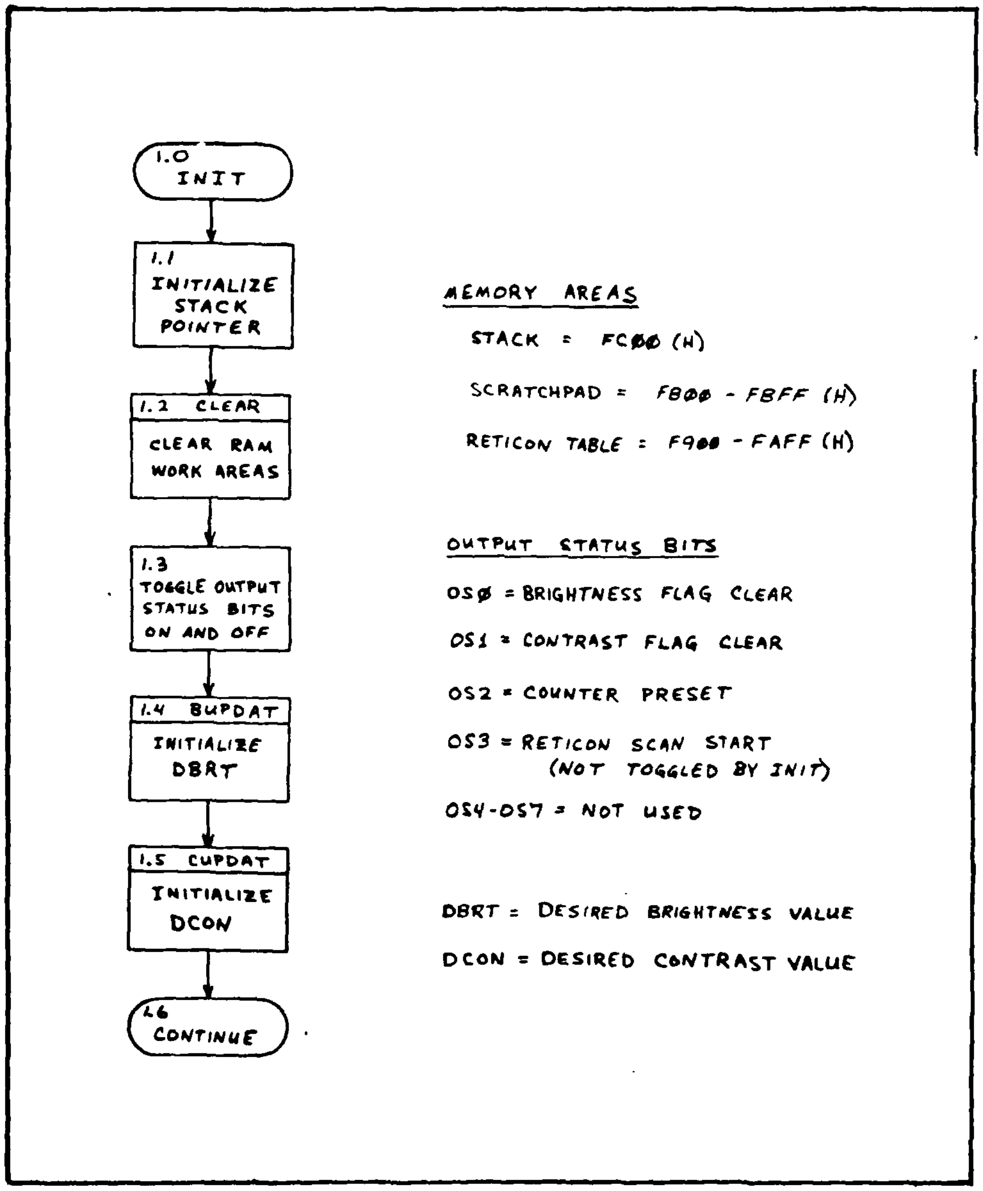
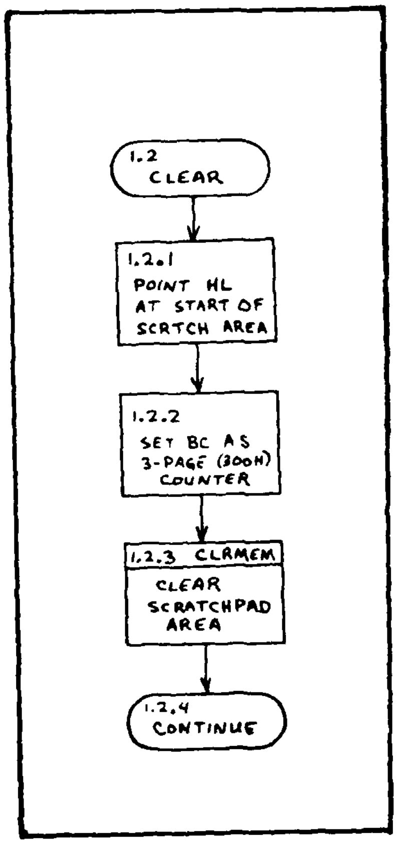

1.0 INIT
1.1 INITIALIZE STACK POINTER
1.2 CLEAR
CLEAR RAM WORK AREAS
1.3 TOGGLE OUTPUT STATUS BITS ON AND OFF
1.4 BUPDAT
INITIALIZE DBRT
1.5 CUPDAT
INITIALIZE DCON
1.6 CONTINUE
MEMORY AREAS
STACK = FCØØ (H)
SCRATCHPAD = FBØØ - FBFF (H)
RETICON TABLE = F9ØØ - FAFF (H)
OUTPUT STATUS BITS
OSØ = BRIGHTNESS FLAG CLEAR
OS1 = CONTRAST FLAG CLEAR
OS2 = COUNTER PRESET
OS3 = RETICON SCAN START (NOT TOGGLED BY INIT)
OS4-OS7 = NOT USED
DBRT = DESIRED BRIGHTNESS VALUE
DCON = DESIRED CONTRAST VALUE

Fig 19. Initialization (INIT) Routine Flowchart



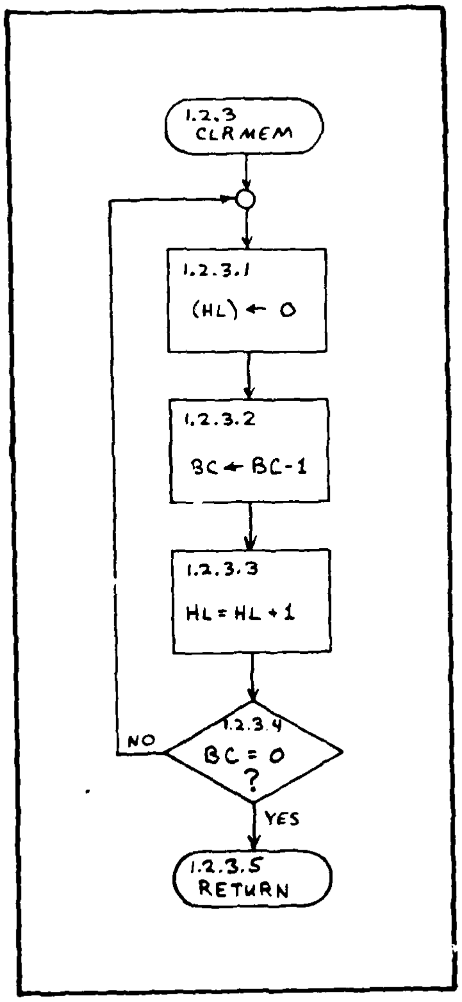


Fig 20. Block Clear Routine Flowchart

Fig 21. Clear Memory (CLRMEM) Routine Flowchart

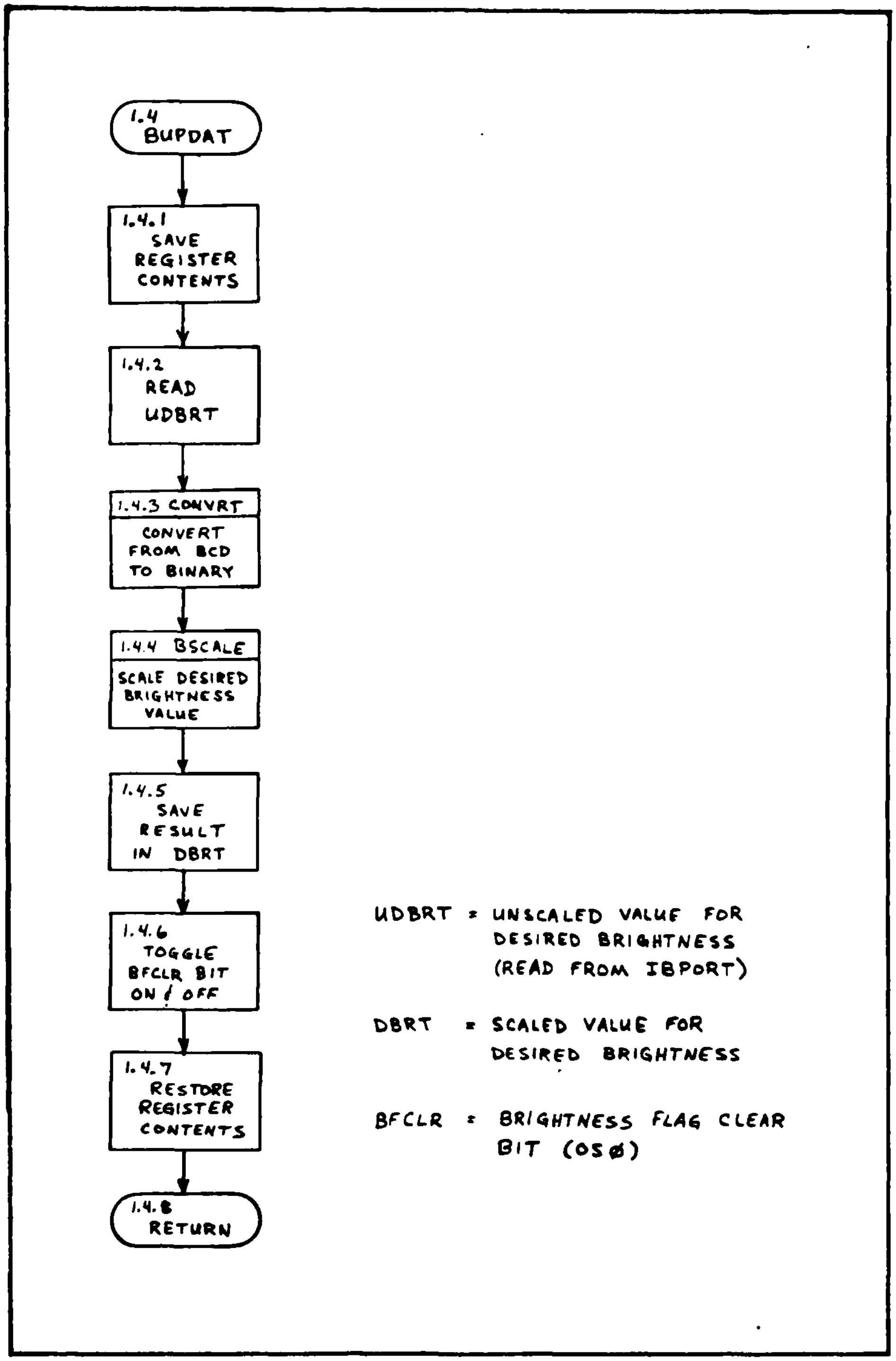
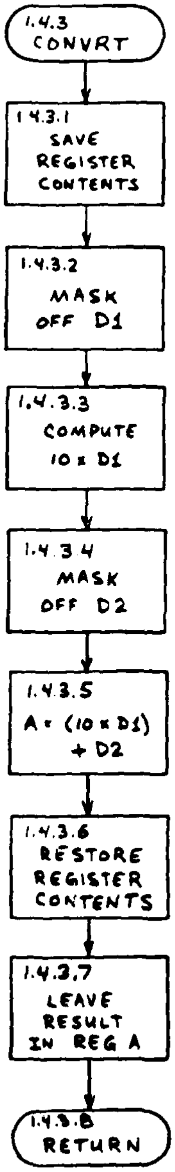

1.4 BUPDAT
1.4.1 SAVE REGISTER CONTENTS
1.4.2 READ UDBRT
1.4.3 CONVRT
CONVERT FROM BCD TO BINARY
1.4.4 BSCALE
SCALE DESIRED BRIGHTNESS VALUE
1.4.5 SAVE RESULT IN DBRT
1.4.6 TOGGLE BFCLR BIT ON / OFF
1.4.7 RESTORE REGISTER CONTENTS
1.4.8 RETURN
UDBRT = UNSCALED VALUE FOR DESIRED BRIGHTNESS (READ FROM IBPORT)
DBRT = SCALED VALUE FOR DESIRED BRIGHTNESS
BFCLR = BRIGHTNESS FLAG CLEAR BIT (OSØ)

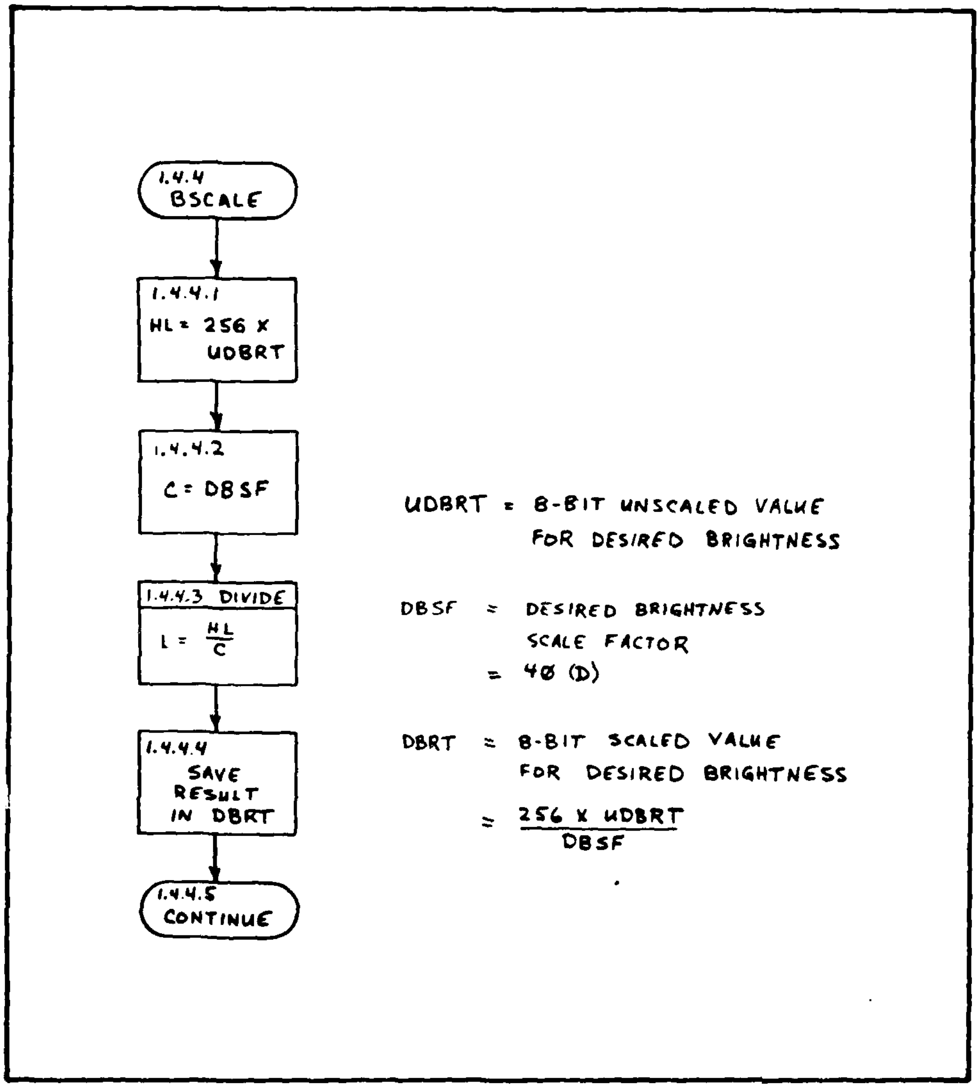
Fig 22. Brightness Update (BUPDAT) Routine Flowchart


1.4.3 CONVRT
1.4.3.1 SAVE REGISTER CONTENTS
1.4.3.2 MASK OFF D1
1.4.3.3 COMPUTE 10 x D1
1.4.3.4 MASK OFF D2
1.4.3.5 A = (10 x D1) + D2
1.4.3.6 RESTORE REGISTER CONTENTS
1.4.3.7 LEAVE RESULT IN REG A
1.4.3.8 RETURN

D1 = MOST SIGNIFICANT FOUR BITS OF BCD BYTE IN REG A

D2 = LEAST SIGNIFICANT FOUR BITS OF BCD BYTE IN REG A

Fig 23. BCD to Binary Conversion (CONVRT) Routine Flowchart



UDBRT = 8-BIT UNSCALED VALUE FOR DESIRED BRIGHTNESS

DBSF = DESIRED BRIGHTNESS SCALE FACTOR
= 4Ø (D)

DBRT = 8-BIT SCALED VALUE FOR DESIRED BRIGHTNESS

$$= \frac{256 \times UDBRT}{DBSF}$$

Fig 24. Brightness Scaling (BSCAL) Routine Flowchart

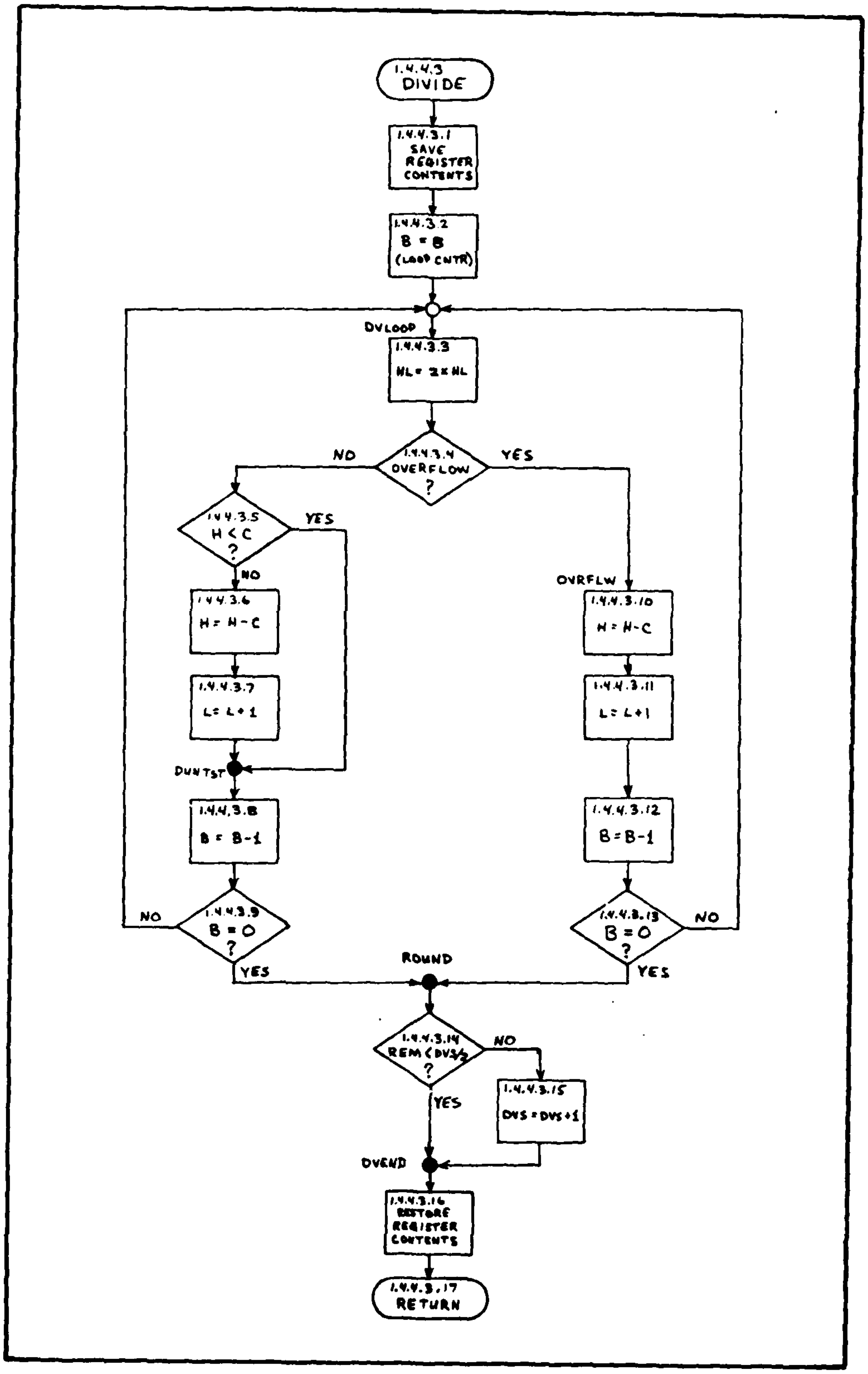
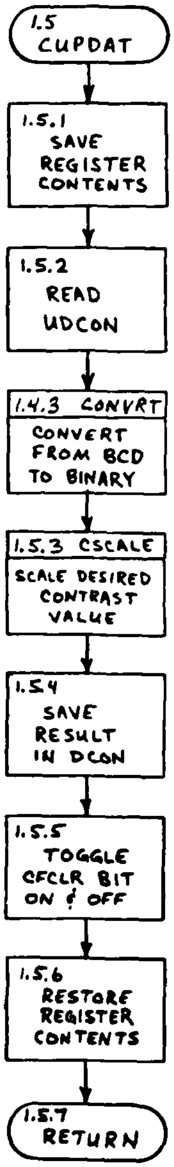


Fig 25. DIVIDE Routine Flowchart

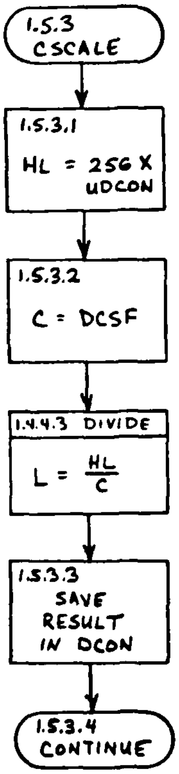


UDCON = UNSCALED VALUE FOR DESIRED CONTRAST (READ FROM ICPORT)

DCON = SCALED VALUE FOR DESIRED CONTRAST

CFCLR = CONTRAST FLAG CLEAR BIT (OS1)

Fig 26. Contrast Update (CUPDAT) Routine Flowchart



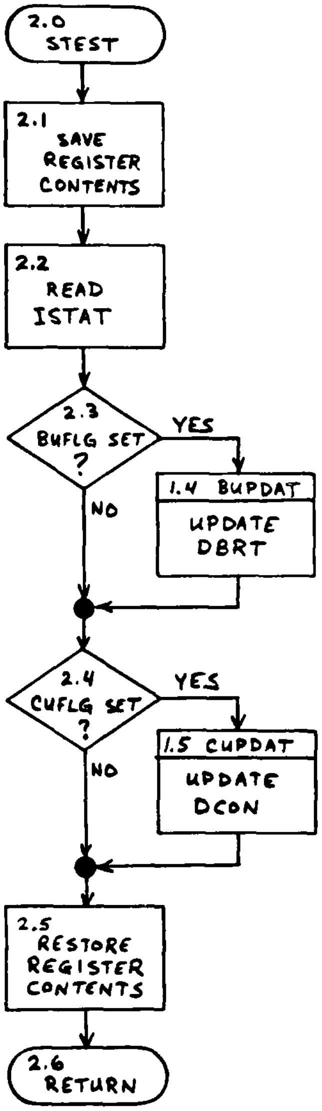
UDCON = 8-BIT UNSCALED VALUE FOR DESIRED CONTRAST

DCSF = DESIRED CONTRAST SCALE FACTOR
= 1ØØ (D)

DCON = 8-BIT SCALED VALUE FOR DESIRED CONTRAST

$$= \frac{256 \times UDCON}{DCSF}$$

Fig 27. Contrast Scaling (CSCAL) Routine Flowchart



ISTAT = INPUT STATUS BYTE (READ FROM ISPORT)

BUFLG = BRIGHTNESS UPDATE FLAG BIT (ISØ)

DBRT = SCALED VALUE FOR DESIRED BRIGHTNESS

CUFLG = CONTRAST UPDATE FLAG BIT (IS1)

DCON = SCALED VALUE FOR DESIRED CONTRAST

Fig 28. Status Update (STEST) Routine Flowchart

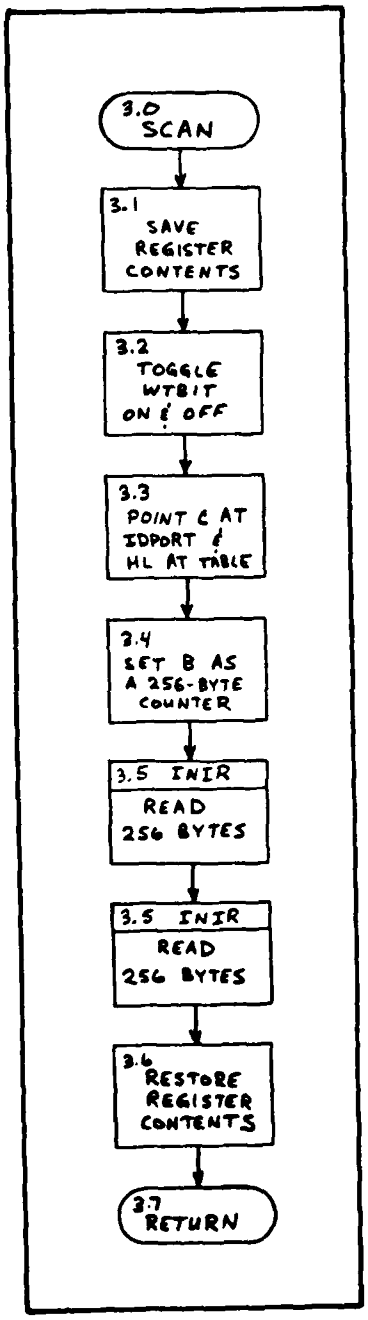

3.0 SCAN
3.1 SAVE REGISTER CONTENTS
3.2 TOGGLE WTBIT ON & OFF
3.3 POINT C AT IDPORT & HL AT TABLE
3.4 SET B AS A 256-BYTE COUNTER
3.5 INIR READ 256 BYTES
3.5 INIR READ 256 BYTES
3.6 RESTORE REGISTER CONTENTS
3.7 RETURN

Fig 29. Reticon SCAN Routine Flowchart

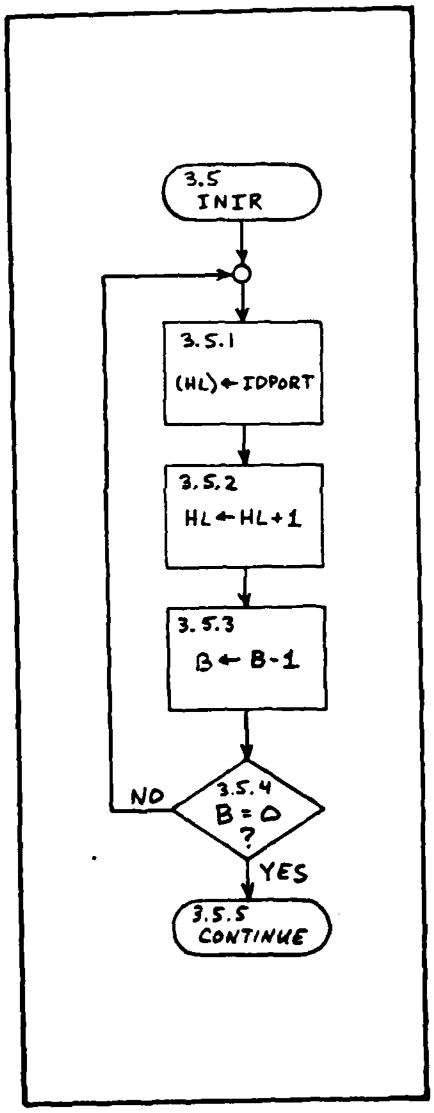
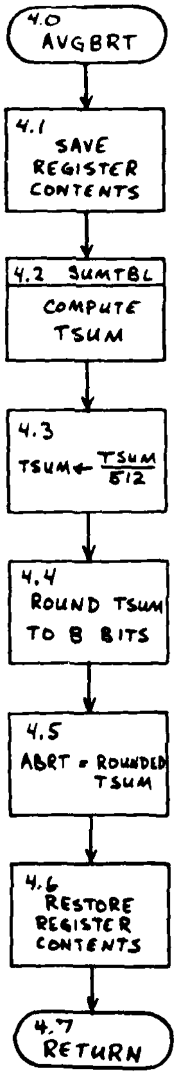


Fig 30. INIR Instruction Flowchart



TSUM = TOTAL SUM OF ALL RETICON SAMPLE VALUES IN RETICON TABLE

ABRT = AVERAGE BRIGHTNESS OF ALL SAMPLE VALUES IN RETICON TABLE

$$= \frac{TSUM}{512}$$

Fig 31. Brightness Calculation (AVGBRT) Routine Flowchart

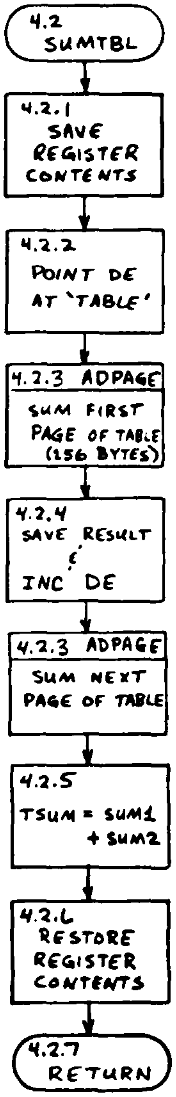


TABLE = START ADDRESS OF RETICON DATA TABLE

SUM1 = SUM OF BYTES IN PAGE1 OF TABLE

SUM2 = SUM OF BYTES IN PAGE2 OF TABLE

TSUM = TOTAL SUM OF ALL RETICON SAMPLE VALUES IN TABLE

Fig 32. Table Summing (SUMTBL) Routine Flowchart

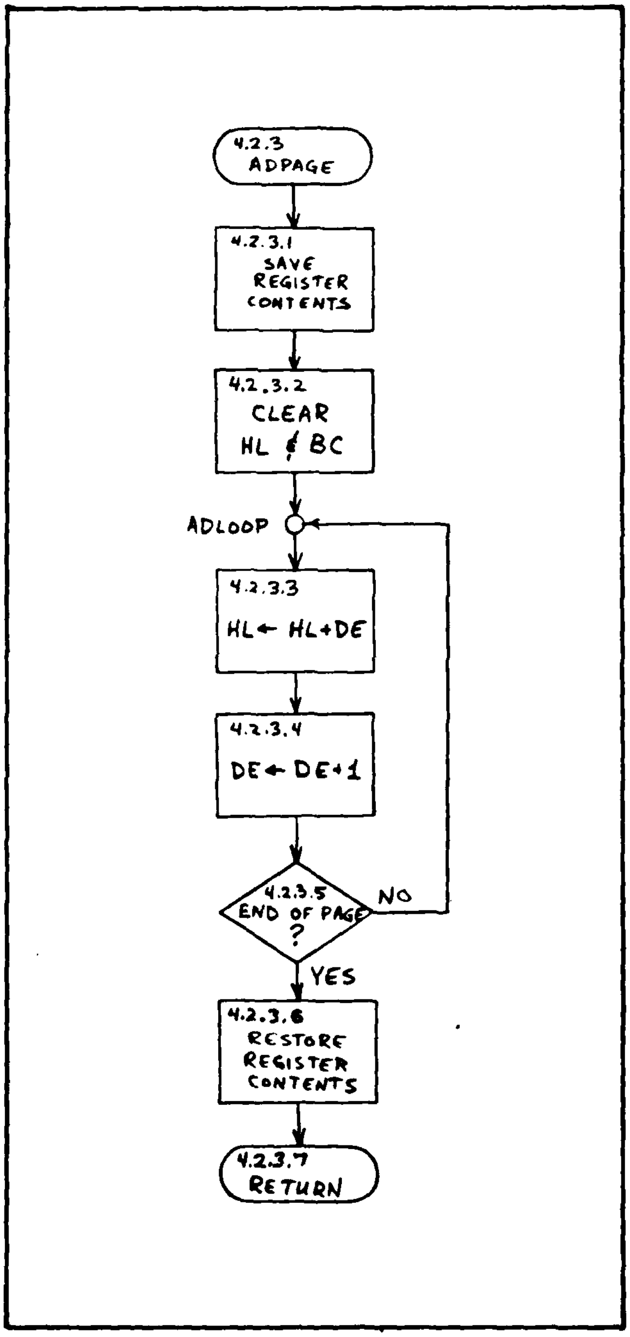
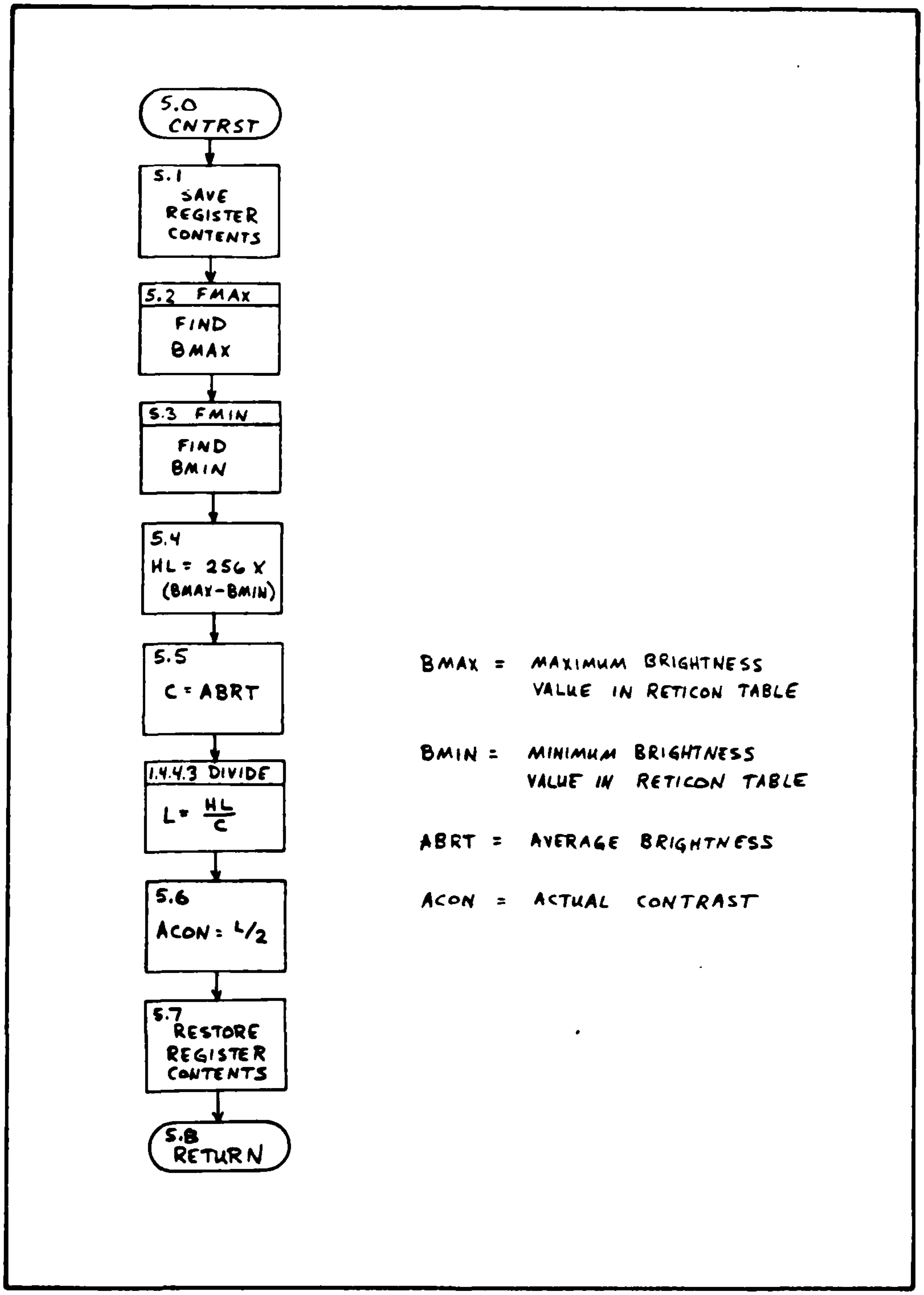


Fig 33. Page Addition (ADPAGE) Routine Flowchart



BMAX = MAXIMUM BRIGHTNESS VALUE IN RETICON TABLE

BMIN = MINIMUM BRIGHTNESS VALUE IN RETICON TABLE

ABRT = AVERAGE BRIGHTNESS

ACON = ACTUAL CONTRAST

Fig 34. Actual Contrast (CONTRST) Routine Flowchart

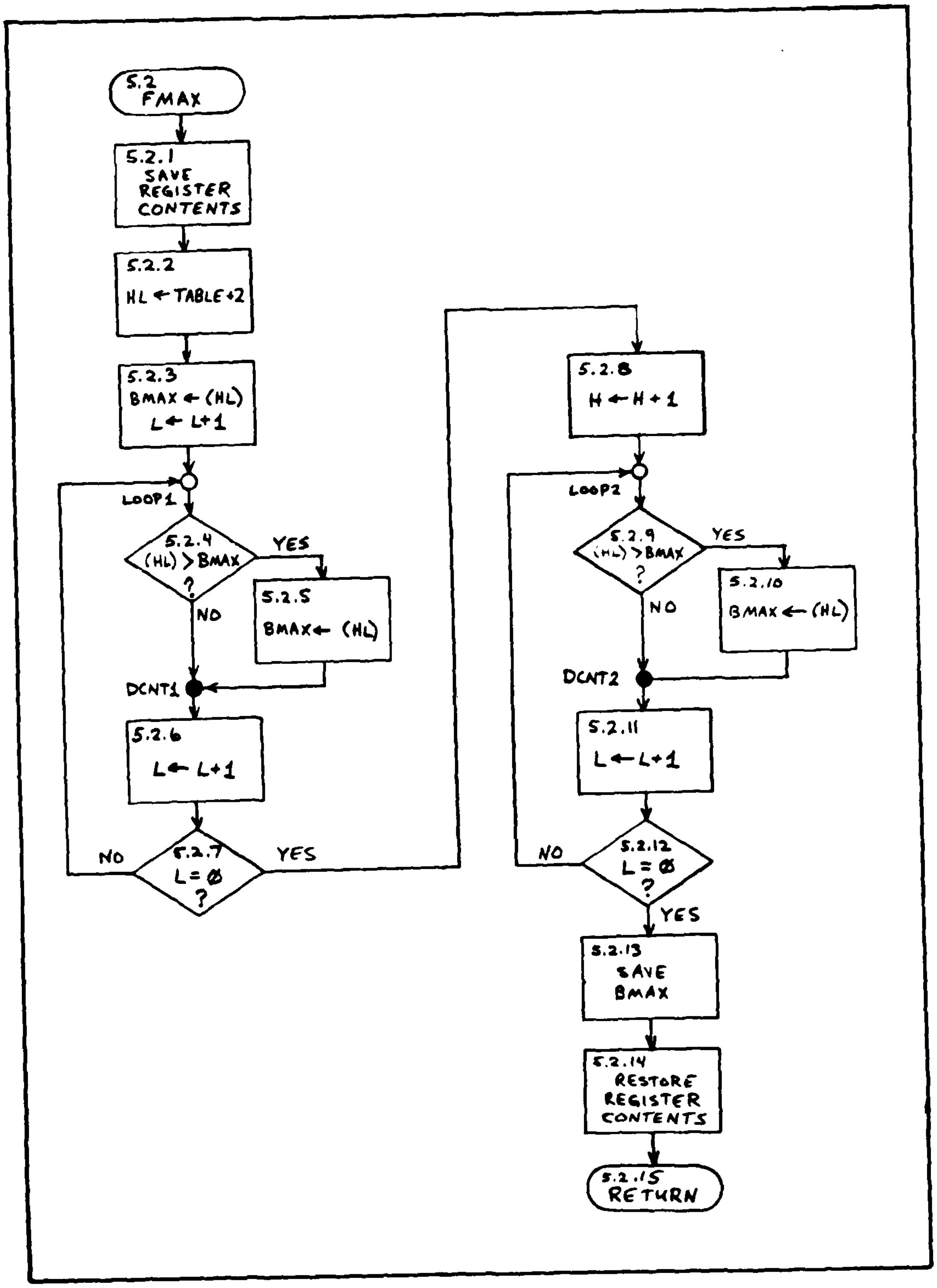


Fig 35. Find Max (FMAX) Routine Flowchart

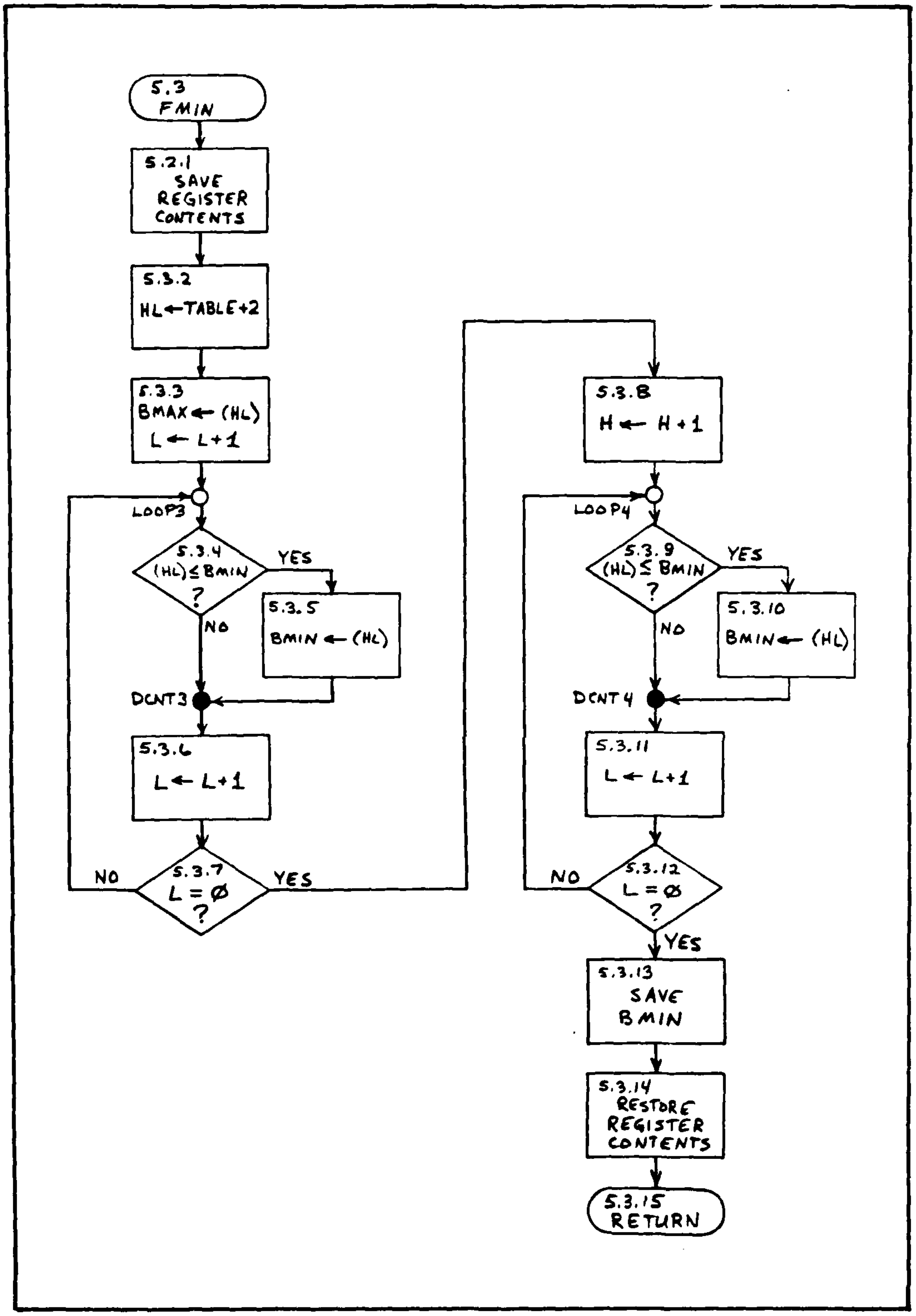
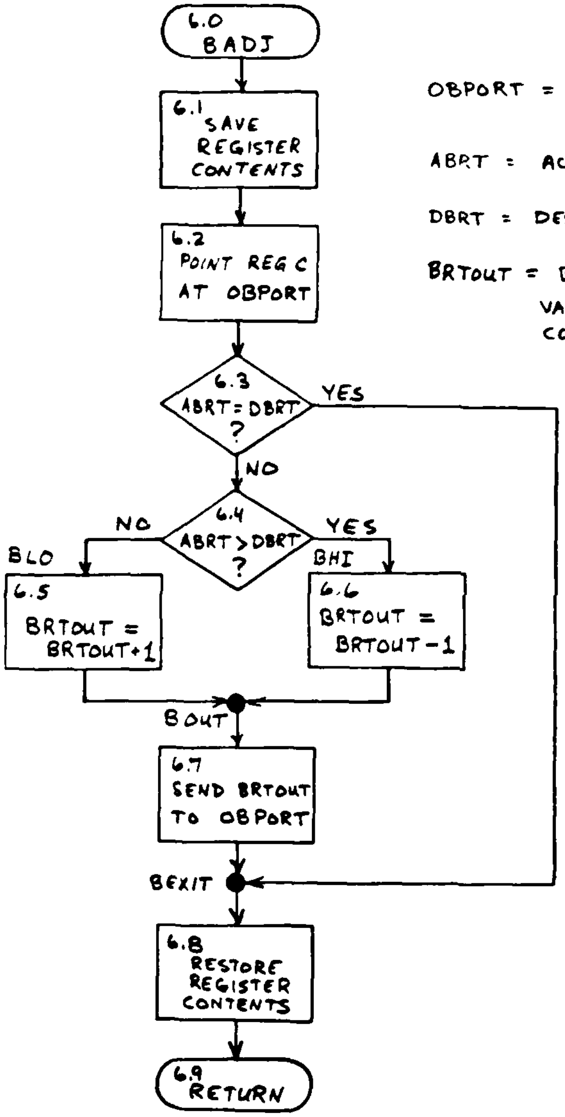


Fig 36. Find Min (FMIN) Routine Flowchart



OBPORT = OUTPUT BRIGHTNESS PORT (OUT PORT 1)

ABRT = ACTUAL BRIGHTNESS VALUE

DBRT = DESIRED BRIGHTNESS VALUE

BRTOUT = BRIGHTNESS OUTPUT VALUE SENT TO BRIGHTNESS CONTROL DAC

Fig 37. Brightness Adjust (BADJ) Routine Flowchart

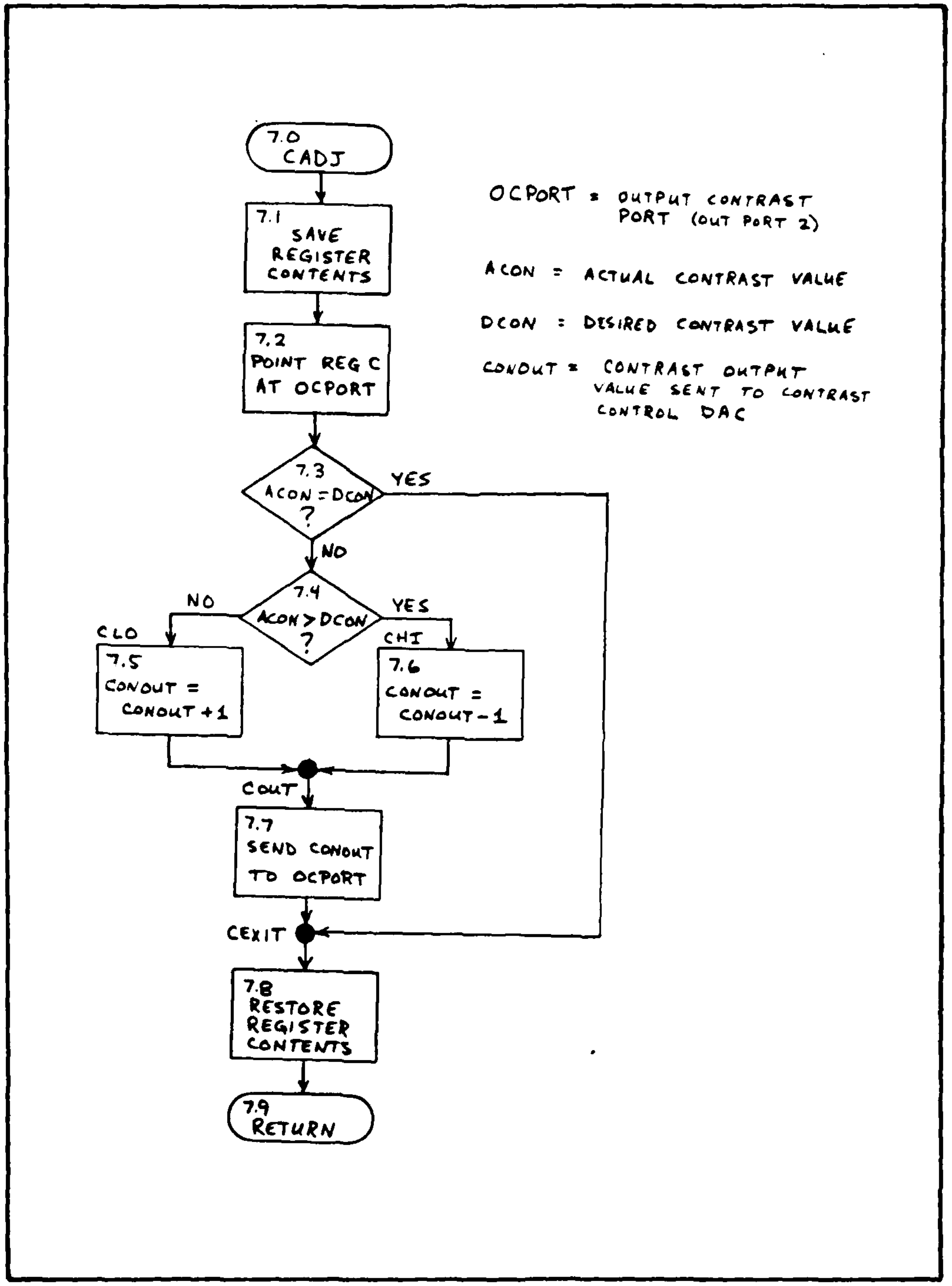

7.0 CADJ
7.1 SAVE REGISTER CONTENTS
7.2 POINT REG C AT OCPORT
7.3 ACON = DCON ?
YES
NO
7.4 ACON > DCON ?
NO
YES
CLO
CHI
7.5 CONOUT = CONOUT + 1
7.6 CONOUT = CONOUT - 1
COUT
7.7 SEND CONOUT TO OCPORT
CEXIT
7.8 RESTORE REGISTER CONTENTS
7.9 RETURN
OCPORT = OUTPUT CONTRAST PORT (OUT PORT 2)
ACON = ACTUAL CONTRAST VALUE
DCON = DESIRED CONTRAST VALUE
CONOUT = CONTRAST OUTPUT VALUE SENT TO CONTRAST CONTROL DAC

Fig 38. Contrast Adjust (CADJ) Routine Flowchart

# APPENDIX D

# ASSEMBLY LANGUAGE PROGRAMS

This appendix contains assembled listings of the software routines described in Appendix B. Appendix C contains flowcharts, which illustrate the algorithms used in these routines.

```
                                        .Z80
0000'                                   CSEG
                                ;TITLE:  SINE-WAVE GRATING CONTROL ROUTINE
                                ;AUTHOR: CAPT B D BAXLEY
                                ;DATE:   30 JAN 83
                                ;SYSTEM: MOSTEK MDX-CPU2
                                ;DESCRIPTION:   THIS PROGRAM IS THE CONTROL SOFTWARE FOR THE
                                ;               SINE-WAVE GRATING CONTROLLER.
                                ;OPERATION:     THE ROUTINES IN THIS PACKAGE HAVE BEEN SET
                                ;               UP TO RUN AS SOON AS THE MDX-CPU2 IS
                                ;               POWERED UP OR RESET (NOTE: THE MDX-CPU2 MUST
                                ;               BE HARD-WIRED TO BEGIN EXECUTION AT 0000H).
                                ;               ONCE RESET, THIS PROGRAM WILL INITIALIZE THE
                                ;               HARDWARE & JUMP INTO AN INFINITE CONTROL LOOP.


                                ;**************** CONSTANTS ************************************

                                ;  MISCELLANEOUS

0000                            BBIT    EQU     00H     ;BRIGHTNESS BIT OF STATUS PORTS
0001                            CBIT    EQU     01H     ;CONTRAST BIT OF STATUS PORTS
0002                            PBIT    EQU     02H     ;PRESET STATUS PORTS BIT
0003                            WTBIT   EQU     03H     ;RESET WAIT STATE CONTROLLER BIT
0028                            DBSF    EQU     40D     ;DESIRED BRIGHTNESS SCALE FACTOR
0064                            DCSF    EQU     100D    ;DESIRED CONTRAST SCALE FACTOR
F800                            SCRTCH  EQU     0F800H  ;BEGINNING OF RAM SCRATCH PAD AREA
FC00                            STACK   EQU     0FC00H  ;TOP OF STACK

                                ;  PORT NUMBERS

0000                            ISPORT  EQU     00H     ;INPUT STATUS PORT
0000                            OSPORT  EQU     00H     ;OUTPUT STATUS PORT
0001                            IBPORT  EQU     01H     ;INPUT BRIGHTNESS PORT
0001                            OBPORT  EQU     01H     ;OUTPUT BRIGHTNESS PORT
0002                            ICPORT  EQU     02H     ;INPUT CONTRAST PORT
0002                            OCPORT  EQU     02H     ;OUTPUT CONTRAST PORT
0003                            IDPORT  EQU     03H     ;INPUT DATA PORT

                                ;*************** VARIABLES ***************************************

F800                            UDBRT   EQU     0F800H  ;UNSCALED BCD VALUE FOR DESIRED BRIGHTNESS
F801                            DBRT    EQU     0F801H  ;8-BIT SCALED VALUE FOR DESIRED BRIGHTNESS
F802                            ABRT    EQU     0F802H  ;COMPUTED ACTUAL BRIGHTNESS
F803                            BRTOUT  EQU     0F803H  ;BRTNESS-OUT SETTING SENT TO OBPORT
F810                            UDCON   EQU     0F810H  ;UNSCALED BCD VALUE FOR DESIRED CONTRAST
F811                            DCON    EQU     0F811H  ;8-BIT SCALED VALUE FOR DESIRED CONTRAST
F812                            ACON    EQU     0F812H  ;COMPUTED ACTUAL CONTRAST
F813                            CONOUT  EQU     0F813H  ;CONTRAST-OUT SETTING SENT TO OCPORT
```

```
  F820                              OSTAT   EQU     0F820H  ;LOCATION OF OUTPUT STATUS BYTE
  F821                              ISTAT   EQU     0F821H  ;LOCATION OF INPUT STATUS BYTE
  F830                              BMAX    EQU     0F830H  ;MAXIMUM SAMPLE VALUE
  F831                              BMIN    EQU     0F831H  ;MINIMUM SAMPLE VALUE
  F832                              TSUM    EQU     0F832H  ;TABLE SUM STORAGE AREA (3 BYTES)
  F900                              TABLE   EQU     0F900H  ;BEGINNING OF 512-BYTE RETICON DATA TBL

                                    ;*************** END VARIABLES **********************************
                                    PAGE
```

```
                                  ;********************************
                                  ;*    INITIALIZATION ROUTINE  - 30 JAN 83          *
                                  ;********************************
                                  ;*                                                  *
                                  ;*  THIS ROUTINE CLEARS ALL RAM WORK AREAS.  THEN THE ROUTINE  *
                                  ;*  INITIALIZES THE CONTROLLER BY STROBING OUTPUT STATUS BITS  *
                                  ;*  ON AND OFF.  THIS PRESETS THE DESIRED BRIGHTNESS AND       *
                                  ;*  CONTRAST COUNTERS AND CLEARS THE FLAG-GENERATING CIRCUITS. *
                                  ;*  BIT 3 OF THE OSPORT IS NOT TOGGLED, HOWEVER, UNTIL THE     *
                                  ;*  PROCESSOR IS READY TO READ A RETICON SCAN.  IN ADDITION,   *
                                  ;*  DESIRED BRIGHTNESS AND CONTRAST VALUES ARE INITIALIZED TO  *
                                  ;*  THE VALUES SPECIFIED BY SWITCHES S5 AND S6.                *
                                  ;*                                                  *
                                  ;*  INPUTS:  NONE                                   *
                                  ;*                                                  *
                                  ;*  OUTPUTS: STACK POINTER IS INITIALIZED           *
                                  ;*           RAM WORK AREAS ARE CLEARED             *
                                  ;*           OSPORT BITS ARE TOGGLED ON AND OFF     *
                                  ;*           DBRT, DCON ARE INITIALIZED             *
                                  ;*                                                  *
                                  ;*  ROUTINES CALLED:  CLRMEM, BUPDAT, CUPDAT        *
                                  ;*                                                  *
                                  ;********************************

                                          ORG     0000H

0000'   31 FC00                   INIT:   LD      SP,STACK        ;INITIALIZE STACK POINTER

0003'   21 F800                           LD      HL,SCRTCH       ;CLEAR SCRATCHPAD WORK AREA
0006'   01 0300                           LD      BC,0300H        ;   AND RETICON DATA TABLE
0009'   CD 0030'                          CALL    CLRMEM

000C'   3E FF                             LD      A,0FFH          ;TOGGLE OUTPUT STATUS BITS
000E'   D3 00                             OUT     (OSPORT),A      ;   ON AND OFF.  (DON'T TOGGLE
0010'   3E 08                             LD      A,008H          ;   BIT 3 TILL READY TO READ
0012'   D3 00                             OUT     (OSPORT),A      ;   RETICON ARRAY)
0014'   32 F820                           LD      (OSTAT),A       ;SAVE OUTPUT STATUS

0017'   CD 0080'                          CALL    BUPDAT          ;INITIALIZE DESIRED BRIGHTNESS
001A'   CD 00B0'                          CALL    CUPDAT          ;INITIALIZE DESIRED CONTRAST

001D'   C3 0040'                          JP      EXEC
                                  PAGE
```

```
                        ;*********************************
                        ;*    CLEAR MEMORY ROUTINE  - 26 JAN 83                 *
                        ;*********************************
                        ;*                                                      *
                        ;*  THIS ROUTINE CLEARS A BLOCK OF MEMORY.  THE ROUTINE WAS    *
                        ;*  TAKEN FROM WADSORTH (REF 13:90).                    *
                        ;*                                                      *
                        ;*  INPUTS:  STARTING ADDRESS OF BLOCK TO CLEAR IN HL   *
                        ;*           NUMBER OF BYTES TO CLEAR IN BC             *
                        ;*                                                      *
                        ;*  OUTPUTS: CLEARS MEMORY BLOCK ADDRESSED BY HL        *
                        ;*                                                      *
                        ;*  REGISTERS AFFECTED:  HL, BC                         *
                        ;*                                                      *
                        ;*  MEMORY LOCATIONS AFFECTED:  (HL) TO (HL + BC - 1)   *
                        ;*                                                      *
                        ;*********************************

                                ORG     0030H

0030'   36 00           CLRMEM: LD      (HL),00H        ;CLEAR MEMORY LOCATION
0032'   ED A1                   CPI                     ;DEC COUNTER, INC HL POINTER
0034'   E0                      RET     PO              ;EXIT IF COUNTER ZERO
0035'   18 F9                   JR      CLRMEM          ;ELSE GO CLEAR NEXT BYTE

                        PAGE
```

```
                                ;*******************************
                                ;*    EXECUTIVE ROUTINE  - 26 JAN 83                 *
                                ;*******************************
                                ;*                                                   *
                                ;*  THIS IS THE MAIN PROGRAM.  ONCE ENTERED, THIS ROUTINE     *
                                ;*  REPEATS ENDLESSLY, CALLING THE OTHER ROUTINES IN THE      *
                                ;*  PROPER ORDER.  THESE ROUTINES ARE DESCRIBED BELOW.        *
                                ;*                                                   *
                                ;*  INPUTS:  NONE                                    *
                                ;*                                                   *
                                ;*  OUTPUTS: SEE INDIVIDUAL ROUTINES BELOW           *
                                ;*                                                   *
                                ;*  ROUTINES CALLED:  STEST, SCAN, AVGBRT, CNTRST, BADJ, CADJ  *
                                ;*                                                   *
                                ;*******************************

                                        ORG     0040H

0040'   CD 0060'                EXEC:   CALL    STEST           ;UPDATE DESIRED BRT & CON

0043'   CD 0160'                        CALL    SCAN            ;READ RETICON SCAN

0046'   CD 0190'                        CALL    AVGBRT          ;COMPUTE AVERAGE BRIGHTNESS

0049'   CD 0200'                        CALL    CNTRST          ;COMPUTE OVERALL CONTRAST

004C'   CD 0290'                        CALL    BADJ            ;CORRECT BRIGHTNESS

004F'   CD 02C0'                        CALL    CADJ            ;CORRECT CONTRAST

0052'   18 EC                           JR      EXEC            ;LOOP FOREVER

                                PAGE
                                ;*********************************
                                ;*    STATUS UPDATE ROUTINE   - 25 JAN 83       *
                                ;*********************************
                                ;*                                               *
                                ;*  THIS ROUTINE TESTS TO SEE IF THE USER WISHES TO UPDATE   *
                                ;*  DESIRED BRIGHTNESS OR CONTRAST VALUES.  IF DESIRED       *
                                ;*  BRIGHTNESS IS TO BE UPDATED, BUPDAT IS CALLED.  IF       *
                                ;*  DESIRED CONTRAST IS TO BE UPDATED, CUPDAT IS CALLED.     *
                                ;*                                               *
                                ;*  INPUTS:  INPUT STATUS PORT IS READ                       *
                                ;*                                               *
                                ;*  OUTPUTS: DBRT AND DCON ARE UPDATED, IF UPDATES NEEDED    *
                                ;*                                               *
                                ;*  ROUTINES CALLED:  BUPDAT, CUPDAT                         *
                                ;*                                               *
                                ;*********************************

                                        ORG     0060H

0060'   F5                      STEST:  PUSH    AF              ;SAVE REGISTER CONTENTS

0061'   DB 00                           IN      A,(ISPORT)      ;CHECK INPUT STATUS BYTE
0063'   32 F821                         LD      (ISTAT),A       ;SAVE IT
0066'   CB 47                           BIT     BBIT,A          ;IF BRIGHTNESS UPDATE REQUESTED,
0068'   C4 0080'                        CALL    NZ,BUPDAT       ;   ...DO IT

006B'   3A F821                         LD      A,(ISTAT)       ;RECALL INPUT STATUS BYTE
006E'   CB 4F                           BIT     CBIT,A          ;IF CONTRAST UPDATE REQUESTED,
0070'   C4 00B0'                        CALL    NZ,CUPDAT       ;   ...DO IT

0073'   F1                              POP     AF              ;RESTORE REGISTERS
0074'   C9                              RET
                                PAGE
                                ;*********************************
                                ;*    BRIGHTNESS UPDATE ROUTINE  - 30 JAN 83           *
                                ;*********************************
                                ;*                                                     *
                                ;*  THIS ROUTINE UPDATES THE VALUE FOR DESIRED BRIGHTNESS.   *
                                ;*  FIRST, THE NEW UNSCALED VALUE FOR DESIRED BRIGHTNESS,    *
                                ;*  UDBRT, IS READ FROM THE IBPORT (INPUT BRIGHTNESS PORT)   *
                                ;*  AND SAVED.  THIS VALUE IS THEN CONVERTED TO A SCALED     *
                                ;*  BINARY VALUE AND SAVED (DBRT = 256 X UDBRT / DBSF).      *
                                ;*  FINALLY, THE ROUTINE TOGGLES BIT 0 OF THE OSPORT ON AND  *
                                ;*  OFF TO SIGNAL THAT BRIGHTNESS HAS BEEN UPDATED.          *
                                ;*                                                     *
                                ;*  INPUTS:  IBPORT IS READ                            *
                                ;*                                                     *
                                ;*  OUTPUTS: UDBRT AND DBRT ARE UPDATED                *
                                ;*           BIT 0 OF OSPORT IS TOGGLED ON AND OFF     *
                                ;*                                                     *
                                ;*  ROUTINES CALLED:  CONVRT, DIVIDE                   *
                                ;*                                                     *
                                ;*********************************

                                        ORG     0080H

0080'   F5                      BUPDAT: PUSH    AF              ;SAVE REGISTER CONTENTS
0081'   E5                              PUSH    HL
0082'   C5                              PUSH    BC

0083'   DB 01                           IN      A,(IBPORT)      ;INPUT NEW VALUE FOR UNSCALED
0085'   32 F800                         LD      (UDBRT),A       ;  DESIRED BRIGHTNESS & SAVE IT
0088'   CD 0100'                        CALL    CONVRT          ;CONVERT IT TO A BINARY VALUE
008B'   67                              LD      H,A             ;LOAD (256 X UDBRT) INTO HL
008C'   2E 00                           LD      L,00H
008E'   0E 28                           LD      C,DBSF          ;LOAD BRTNESS SCALE FACTOR IN C
0090'   CD 0120'                        CALL    DIVIDE          ;CALC DBRT = 256 X UDBRT / DBSF
0093'   7D                              LD      A,L             ;SAVE RESULT
0094'   32 F801                         LD      (DBRT),A

0097'   3A F820                         LD      A,(OSTAT)       ;TOGGLE BUPDAT ACKNOWLEDGE BIT
009A'   CB C7                           SET     BBIT,A          ;   ... ON ...
009C'   D3 00                           OUT     (OSPORT),A
009E'   CB 87                           RES     BBIT,A          ;   ... THEN OFF
00A0'   D3 00                           OUT     (OSPORT),A

00A2'   C1                              POP     BC              ;RESTORE REGISTERS
00A3'   E1                              POP     HL
00A4'   F1                              POP     AF
00A5'   C9                              RET
                                PAGE
```

```
                              ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                              ; *    CONTRAST UPDATE ROUTINE  - 30 JAN 83                       *
                              ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                              ; *                                                               *
                              ; *  THIS ROUTINE UPDATES THE VALUE FOR DESIRED CONTRAST.         *
                              ; *  FIRST, THE NEW UNSCALED VALUE FOR DESIRED CONTRAST,          *
                              ; *  UDCON, IS READ FROM THE ICPORT (INPUT CONTRAST PORT)         *
                              ; *  AND SAVED.  THIS VALUE IS THEN CONVERTED TO A SCALED         *
                              ; *  BINARY VALUE AND SAVED (DCON = 256 X UDCON / DCSF).          *
                              ; *  FINALLY, THE ROUTINE TOGGLES BIT 1 OF THE OSPORT ON AND      *
                              ; *  OFF TO SIGNAL THAT CONTRAST HAS BEEN UPDATED.                *
                              ; *                                                               *
                              ; *  INPUTS:  ICPORT IS READ                                      *
                              ; *                                                               *
                              ; *  OUTPUTS: UDCON AND DCON ARE UPDATED                          *
                              ; *           BIT 1 OF OSPORT IS TOGGLED ON AND OFF               *
                              ; *                                                               *
                              ; *  ROUTINES CALLED:  CONVRT, DIVIDE                             *
                              ; *                                                               *
                              ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                                      ORG     00B0H

00B0'   F5                    CUPDAT: PUSH    AF              ;SAVE REGISTER CONTENTS
00B1'   E5                            PUSH    HL
00B2'   C5                            PUSH    BC

00B3'   DB 02                         IN      A,(ICPORT)      ;INPUT NEW VALUE FOR UNSCALED
00B5'   32 F810                       LD      (UDCON),A       ;   DESIRED CONTRAST & SAVE IT
00B8'   CD 0100'                      CALL    CONVRT          ;CONVERT IT TO A BINARY VALUE
00BB'   67                            LD      H,A             ;LOAD (256 X UDCON) INTO HL
00BC'   2E 00                         LD      L,00H
00BE'   0E 64                         LD      C,DCSF          ;LOAD CONTRAST SCALE FACTOR IN C
00C0'   CD 0120'                      CALL    DIVIDE          ;CALC DCON = 256 X UDCON / DCSF
00C3'   7D                            LD      A,L             ;SAVE RESULT
00C4'   32 F811                       LD      (DCON),A

00C7'   3A F820                       LD      A,(OSTAT)       ;TOGGLE CUPDAT ACKNOWLEDGE BIT
00CA'   CB CF                         SET     CBIT,A          ;   ... ON ...
00CC'   D3 00                         OUT     (OSPORT),A
00CE'   CB 8F                         RES     CBIT,A          ;   ... THEN OFF
00D0'   D3 00                         OUT     (OSPORT),A

00D2'   C1                            POP     BC              ;RESTORE REGISTERS
00D3'   E1                            POP     HL
00D4'   F1                            POP     AF
00D5'   C9                            RET
                              PAGE
```

```
                                ;********************************************
                                ;*    DECIMAL TO BINARY CONVERSION ROUTINE    - 30 JAN 83     *
                                ;********************************************
                                ;*                                                            *
                                ;*  THIS ROUTINE CONVERTS AN 8-BIT BINARY-CODED DECIMAL       *
                                ;*  (BCD) NUMBER TO ITS BINARY EQUIVALENT                     *
                                ;*                                                            *
                                ;*  INPUTS:  8-BIT BCD NUMBER IN A                            *
                                ;*                                                            *
                                ;*  OUTPUTS: 8-BIT BINARY EQUIVALENT IN A                     *
                                ;*                                                            *
                                ;*  REGISTERS AFFECTED:  A                                    *
                                ;*                                                            *
                                ;********************************************

                                        ORG     0100H

0100'   D5                      CONVRT: PUSH    DE              ;SAVE REGISTER CONTENTS
0101'   F5                              PUSH    AF

0102'   5F                              LD      E,A             ;SAVE THE DECIMAL NUMBER IN E
0103'   E6 F0                           AND     0F0H            ;STRIP OFF DIGIT2 & CLEAR C BIT
0105'   1F                              RRA                     ;A = 8 X D1
0106'   57                              LD      D,A             ;SAVE 8 X D1
0107'   1F                              RRA                     ;A = 4 X D1
0108'   1F                              RRA                     ;A = 2 X D1
0109'   82                              ADD     A,D             ;A = 10 X D1
010A'   57                              LD      D,A             ;SAVE 10 X D1

010B'   7B                              LD      A,E             ;GET DIGIT2 OF THE DECIMAL NO.
010C'   E6 0F                           AND     0FH             ;STRIP OFF DIGIT1
010E'   82                              ADD     A,D             ;A = (10 X D1) + D2
010F'   57                              LD      D,A             ;SAVE RESULT

0110'   F1                              POP     AF              ;RESTORE F REGISTER
0111'   7A                              LD      A,D             ;LEAVE BINARY VALUE IN A
0112'   D1                              POP     DE              ;RESTORE DE
0113'   C9                              RET
                                PAGE
                                ;*********************************
                                ;*     DIVIDE ROUTINE  - 29 JAN 83                   *
                                ;*********************************
                                ;*                                                   *
                                ;*  THIS ROUTINE DIVIDES A 16-BIT UNSIGNED DIVIDEND IN HL BY  *
                                ;*  AN 8-BIT UNSIGNED DIVISOR IN C TO YIELD AN 8-BIT QUOTIENT *
                                ;*  (WITH LSB ROUNDED) IN L.  THE REMAINDER THAT EXISTED      *
                                ;*  BEFORE ROUNDING IS LEFT IN H.                            *
                                ;*                                                   *
                                ;*  INPUTS:  16-BIT DIVIDEND (DVD) IN HL                     *
                                ;*           8-BIT DIVISOR (DVS) IN C                        *
                                ;*                                                   *
                                ;*  OUTPUTS: 8-BIT ROUNDED QUOTIENT (QUO) IN L               *
                                ;*           8-BIT REMAINDER (BEFORE ROUNDING) IN H          *
                                ;*                                                   *
                                ;*  REG'S AFFECTED: HL                                       *
                                ;*                                                   *
                                ;*********************************

                                        ORG     0120H

0120'   F5                      DIVIDE: PUSH    AF              ;SAVE REGISTER CONTENTS
0121'   C5                              PUSH    BC
0122'   06 08                           LD      B,08H           ;INITIALIZE COUNTER FOR 8 PASSES

0124'   29                      DVLOOP: ADD     HL,HL           ;SHIFT DVD & QUO LEFT 1 BIT
0125'   38 0A                           JR      C,OVRFLW        ;JUMP IF DVD OVERFLOWED
0127'   7C                              LD      A,H
0128'   91                              SUB     C               ;CAN DVD BE DIVIDED?
0129'   38 02                           JR      C.DUNTST        ;IF NO, GO TO NEXT STEP
012B'   67                              LD      H,A             ;IF YES, DVD - DVS
012C'   2C                              INC     L               ;SET LSB OF QUO
012D'   10 F5                   DUNTST: DJNZ    DVLOOP          ;LOOP UNTIL CTR = 0
012F'   18 06                           JR      ROUND           ;GO ROUND RESULT IF DONE

0131'   7C                      OVRFLW: LD      A,H             ;GET DVD
0132'   91                              SUB     C               ;DVD - DVS
0133'   67                              LD      H,A             ;SAVE RESULT IN DVD
0134'   2C                              INC     L               ;SET LSB OF QUO
0135'   10 ED                           DJNZ    DVLOOP          ;LOOP UNTIL CTR = 0

0137'   A7                      ROUND:  AND     A               ;CLEAR CARRY BIT
0138'   79                              LD      A,C             ;DIVIDE DVS BY 2
0139'   1F                              RRA
013A'   BC                              CP      H               ;COMPARE REM WITH 1/2 DVS
013B'   30 01                           JR      NC,DVEND        ;IF REM ( 1/2 DVS, RETURN
013D'   2C                              INC     L               ;ELSE ROUND QUO UP 1 BIT
013E'   C1                      DVEND:  POP     BC              ;RESTORE REGISTERS
013F'   F1                              POP     AF
0140'   C9                              RET
                                PAGE
                                ;********************************
                                ;*    SCAN ROUTINE    -  25 JAN 83                  *
                                ;********************************
                                ;*                                                  *
                                ;*  THIS ROUTINE READS 512 BYTES OF DATA FROM PORT 3.  DATA    *
                                ;*  IS READ AS QUICKLY AS POSSIBLE, WITH NO WAITING FOR        *
                                ;*  HANDSHAKES.  IT IS ASSUMED THAT THE EXTERNAL HARDWARE      *
                                ;*  RUNS SLOWER THAN THIS ROUTINE, AND INSERTS WAIT STATES     *
                                ;*  TO SYNCHRONIZE THE COMPUTER WITH THE OTHER HARDWARE.       *
                                ;*                                                  *
                                ;*  INPUTS:  INPUT PORT 3 IS READ                   *
                                ;*                                                  *
                                ;*  OUTPUTS: 512-BYTE RETICON DATA TABLE IS UPDATED            *
                                ;*           WTBIT OF OUTPUT STATUS PORT IS TOGGLED OFF & ON   *
                                ;*                                                  *
                                ;********************************

                                        ORG     0160H

0160'   F5                      SCAN:   PUSH    AF              ;SAVE REGISTER CONTENTS
0161'   E5                              PUSH    HL
0162'   C5                              PUSH    BC

0163'   3A F820                         LD      A,(OSTAT)       ;RESET WAIT STATE CONTROLLER
0166'   CB 9F                           RES     WTBIT,A         ;   BY TOGGLING RESET BIT OFF ...
0168'   D3 00                           OUT     (OSPORT),A
016A'   CB DF                           SET     WTBIT,A         ; ... THEN ON AGAIN.
016C'   D3 00                           OUT     (OSPORT),A

016E'   0E 03                           LD      C,IDPORT        ;AIM C AT INPUT DATA PORT
0170'   21 F900                         LD      HL,TABLE        ;AIM HL AT START OF TABLE
0173'   06 00                           LD      B,00H           ;SET B AS A 256-BYTE COUNTER

0175'   ED B2                           INIR                    ;READ 256 BYTES
0177'   ED B2                           INIR                    ;READ NEXT 256 BYTES

0179'   C1                              POP     BC              ;RESTORE REGISTERS
017A'   E1                              POP     HL
017B'   F1                              POP     AF
017C'   C9                              RET
                                PAGE
                                ;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                                ;*    BRIGHTNESS CALCULATION ROUTINE  -  30 JAN 83               *
                                ;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                                ;*                                                               *
                                ;*  THIS ROUTINE COMPUTES AVERAGE BRIGHTNESS MEASURED BY THE     *
                                ;*  RETICON ARRAY.  ALL 512 VALUES ARE SUMMED AND THE RESULT     *
                                ;*  IS ROUNDED TO 8 BITS (i.e. DIVIDED BY 512)                   *
                                ;*                                                               *
                                ;*  INPUTS:  RETICON DATA                                        *
                                ;*                                                               *
                                ;*  OUTPUTS: ABRT (ACTUAL BRIGHTNESS) IS UPDATED                 *
                                ;*                                                               *
                                ;*  ROUTINES CALLED:  SUMTBL                                     *
                                ;*                                                               *
                                ;*  MEMORY LOCNS AFFECTED:  ABRT, TSUM                           *
                                ;*                                                               *
                                ;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                                        ORG     0190H

0190'   F5                      AVGBRT: PUSH    AF              ;SAVE REGISTER CONTENTS
0191'   E5                              PUSH    HL

0192'   CD 01B0'                        CALL    SUMTBL          ;GO ADD ALL TABLE VALUES

0195'   2A F833                         LD      HL,(TSUM+1)     ;ROUND 8 MOST SIGNIFICANT BITS
0198'   CB 45                           BIT     0,L             ;  OF TSUM UP OR DOWN
019A'   28 01                           JR      Z,SHIFT         ;IF BIT 0 NOT SET, SKIP AHEAD
019C'   23                              INC     HL              ;IF BIT 0 SET, ROUND UP

019D'   CB 1C                   SHIFT:  RR      H               ;SHIFT MOST SIGNIFICANT 8 BITS
019F'   CB 1D                           RR      L               ;   INTO L.  THIS EFFECTIVELY
01A1'   7D                              LD      A,L             ;   DIVIDES TSUM BY 512
01A2'   32 F802                         LD      (ABRT),A        ;SAVE RESULT

01A5'   E1                              POP     HL              ;RESTORE REGISTERS
01A6'   F1                              POP     AF
01A7'   C9                              RET                     ;RETURN
                                PAGE
```

```
                             ;*********************************
                             ;*    TABLE SUMMING ROUTINE   -  30 JAN 83         *
                             ;*********************************
                             ;*                                                 *
                             ;*  THIS ROUTINE SUMS ALL 512 BYTES STORED IN THE DATA TABLE.  *
                             ;*  THE RESULT IS STORED IN THREE SEQUENTIAL MEMORY LOCATIONS  *
                             ;*  ADDRESSED BY TSUM, WITH THE LSB IN THE FIRST BYTE.         *
                             ;*                                                 *
                             ;*  INPUTS:  RETICON DATA                          *
                             ;*                                                 *
                             ;*  OUTPUTS: TSUM IS UPDATED (SUM OF RETICON DATA SAMPLES)     *
                             ;*                                                 *
                             ;*  ROUTINES CALLED:  ADPAGE                       *
                             ;*                                                 *
                             ;*  MEMORY LOCNS AFFECTED:  TSUM (3 BYTES)         *
                             ;*                                                 *
                             ;*********************************

                                     ORG     01B0H

01B0'   F5                   SUMTBL: PUSH    AF              ;SAVE REGISTER CONTENTS
01B1'   E5                           PUSH    HL
01B2'   D5                           PUSH    DE
01B3'   C5                           PUSH    BC

01B4'   11 F900                      LD      DE,TABLE        ;AIM DE AT BEGINNING OF TABLE
01B7'   CD 01E0'                     CALL    ADPAGE          ;GO SUM FIRST 256 DATA VALUES

01BA'   44                           LD      B,H             ;SAVE PAGE1 SUM IN BC
01BB'   4D                           LD      C,L
01BC'   14                           INC     D               ;AIM DE AT PAGE2 OF RETICON DATA
01BD'   CD 01E0'                     CALL    ADPAGE          ;GO ADD NEXT 256 BYTES

01C0'   09                           ADD     HL,BC           ;ADD PAGE1 SUM TO PAGE2 SUM
01C1'   22 F832                      LD      (TSUM),HL       ;SAVE RESULT IN TSUM
01C4'   3E 00                        LD      A,00H           ;SAVE ANY OVERFLOW (i.e. CARRY
01C6'   17                           RLA                     ;   BIT) IN TSUM'S MSB
01C7'   32 F834                      LD      (TSUM+2),A

01CA'   C1                   STEXIT: POP     BC              ;RESTORE REGISTERS
01CB'   D1                           POP     DE
01CC'   E1                           POP     HL
01CD'   F1                           POP     AF
01CE'   C9                           RET

                             PAGE
```

```
                            ;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                            ;*    PAGE ADDITION ROUTINE   -  30 JAN 83                        *
                            ;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                            ;*                                                                *
                            ;*  THIS ROUTINE SUMS THE MEMORY PAGE (256 BYTES) ADDRESSED       *
                            ;*  BY DE.  THE PAGE SUM IS LEFT IN HL.                           *
                            ;*                                                                *
                            ;*  INPUTS:  ADDRESS OF PAGE'S FIRST BYTE IN DE                   *
                            ;*                                                                *
                            ;*  OUTPUTS: SUM OF ALL VALUES IN PAGE IN HL                      *
                            ;*                                                                *
                            ;*  REGISTERS AFFECTED: HL                                        *
                            ;*                                                                *
                            ;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                                    ORG     01E0H

01E0'   F5                  ADPAGE: PUSH    AF              ;SAVE REGISTER CONTENTS
01E1'   D5                          PUSH    DE
01E2'   C5                          PUSH    BC

01E3'   21 0000                     LD      HL,00H          ;CLEAR BOTH REGISTER PAIRS
01E6'   01 0000                     LD      BC,00H          ;   USED FOR ADDITION

01E9'   1A                  ADLOOP: LD      A,(DE)          ;GET BYTE TO BE ADDED
01EA'   4F                          LD      C,A
01EB'   09                          ADD     HL,BC           ;ADD IT TO CURRENT SUM
01EC'   1C                          INC     E               ;AIM DE AT NEXT BYTE
01ED'   20 FA                       JR      NZ,ADLOOP       ;IF NOT DONE, GO ADD NEXT BYTE

01EF'   C1                          POP     BC              ;RESTORE REGISTERS
01F0'   D1                          POP     DE
01F1'   F1                          POP     AF
01F2'   C9                          RET
                            PAGE
```

```
                                ;*********************************
                                ;*    ACTUAL CONTRAST ROUTINE  - 30 JAN 83       *
                                ;*********************************
                                ;*                                               *
                                ;*  THIS ROUTINE COMPUTES ACTUAL CONTRAST FROM THE RETICON    *
                                ;*  DATA.  THE ALGORITHM USED IS:                 *
                                ;*           ACON = 256 X (BMAX - BMIN) / (BMAX + BMIN)     *
                                ;*                = 256 X (BMAX - BMIN) / (2 X BAVG)        *
                                ;*  NOTE:  ACON IS ACTUALLY SCALED (MULTIPLIED BY 256)      *
                                ;*         SINCE TRUE ACON WOULD BE A FRACTION.             *
                                ;*                                               *
                                ;*  INPUTS:  ABRT (= BAVG), RETICON DATA          *
                                ;*                                               *
                                ;*  OUTPUTS: ACON, BMAX, BMIN ARE UPDATED         *
                                ;*                                               *
                                ;*  ROUTINES CALLED:  FMAX, FMIN, DIVIDE          *
                                ;*                                               *
                                ;*********************************

                                        ORG     0200H

0200'   F5                      CNTRST: PUSH    AF              ;SAVE REGISTER CONTENTS
0201'   E5                              PUSH    HL
0202'   C5                              PUSH    BC

0203'   CD 0230'                        CALL    FMAX            ;FIND MAX BRIGHTNESS VALUE
0206'   CD 0260'                        CALL    FMIN            ;FIND MIN BRIGHTNESS VALUE
0209'   3A F831                         LD      A,(BMIN)        ;LOAD MIN BRIGHTNESS
020C'   47                              LD      B,A
020D'   3A F830                         LD      A,(BMAX)        ;LOAD MAX BRIGHTNESS
0210'   90                              SUB     B               ;BMAX - BMIN
0211'   67                              LD      H,A             ;PUT DIVIDEND OF
0212'   2E 00                           LD      L,00H           ;   256 X (BMAX - BMIN) IN HL
0214'   3A F802                         LD      A,(ABRT)        ;PUT DIVISOR OF
0217'   4F                              LD      C,A             ;   ABRT (= BAVG) IN C
0218'   CD 0120'                        CALL    DIVIDE          ;COMPUTE 256 X (BMAX-BMIN)/ABRT
021B'   A7                              AND     A               ;CLEAR CARRY BIT
021C'   7D                              LD      A,L             ;NOW COMPUTE
021D'   1F                              RRA                     ;   256 X (BMAX-BMIN)/(2 X ABRT)
021E'   32 F812                         LD      (ACON),A        ;SAVE RESULT

0221'   C1                              POP     BC              ;RESTORE REGISTERS
0222'   E1                              POP     HL
0223'   F1                              POP     AF
0224'   C9                              RET
                                PAGE
                                ;*******************************************
                                ;*     FIND MAX ROUTINE  - 26 JAN 83      *
                                ;*******************************************
                                ;*                                         *
                                ;*  THIS ROUTINE FINDS THE MAX VALUE IN THE RETICON DATA TABLE *
                                ;*                                         *
                                ;*  INPUTS:  RETICON DATA                  *
                                ;*                                         *
                                ;*  OUTPUTS: BMAX IS UPDATED               *
                                ;*                                         *
                                ;*******************************************

                                        ORG     0230H

0230'   F5                      FMAX:   PUSH    AF              ;SAVE REGISTER CONTENTS
0231'   E5                              PUSH    HL
0232'   C5                              PUSH    BC

0233'   21 F902                         LD      HL,TABLE+2      ;AIM TABLE POINTER AT FIRST
                                                                ;SAMPLE OF INTEREST (SKIP 1ST TWO)

                                                                ;*** FIND MAX IN FIRST PAGE ***
0236'   7E                              LD      A,(HL)          ;SAVE 1ST VALUE AS MAX
0237'   2C                              INC     L               ;INCREMENT POINTER
0238'   BE                      LOOP1:  CP      (HL)            ;COMPARE NEXT VALUE
0239'   30 01                           JR      NC,DCNT1        ;IF NOT LARGER, SKIP AHEAD
023B'   7E                              LD      A,(HL)          ;IF LARGER, SAVE NEW MAX
023C'   2C                      DCNT1:  INC     L               ;INCREMENT POINTER
023D'   20 F9                           JR      NZ,LOOP1        ;LOOP TILL DONE WITH PAGE1

023F'   24                              INC     H               ;***DO THE SAME WITH 2ND PAGE***
0240'   BE                      LOOP2:  CP      (HL)
0241'   30 01                           JR      NC,DCNT2
0243'   7E                              LD      A,(HL)
0244'   2C                      DCNT2:  INC     L
0245'   20 F9                           JR      NZ,LOOP2
0247'   32 F830                         LD      (BMAX),A        ;SAVE FINAL BMAX VALUE

024A'   C1                              POP     BC              ;RESTORE REGISTERS
024B'   E1                              POP     HL
024C'   F1                              POP     AF
024D'   C9                              RET

                                PAGE
```

```
                                 ;*********************************
                                 ;*    FIND MIN ROUTINE  - 26 JAN 83             *
                                 ;*********************************
                                 ;*                                              *
                                 ;*  THIS ROUTINE FINDS THE MIN VALUE IN THE RETICON DATA TABLE *
                                 ;*                                              *
                                 ;*  INPUTS:  RETICON DATA                       *
                                 ;*                                              *
                                 ;*  OUTPUTS: BMIN IS UPDATED                    *
                                 ;*                                              *
                                 ;*********************************

                                         ORG     0260H

0260'   F5                       FMIN:   PUSH    AF              ;SAVE REGISTER CONTENTS
0261'   E5                               PUSH    HL
0262'   C5                               PUSH    BC

0263'   21 F902                          LD      HL,TABLE+2      ;AIM TABLE POINTER AT FIRST
                                                                 ;SAMPLE OF INTEREST (SKIP 1ST TWO)

                                                                 ;*** FIND MIN IN FIRST PAGE ***
0266'   7E                               LD      A,(HL)          ;SAVE 1ST VALUE AS MIN
0267'   2C                               INC     L               ;INCREMENT POINTER
0268'   BE                       LOOP3:  CP      (HL)            ;COMPARE NEXT VALUE
0269'   38 01                            JR      C,DCNT3         ;IF LARGER, SKIP AHEAD
026B'   7E                               LD      A,(HL)          ;   ELSE, SAVE NEW MIN
026C'   2C                       DCNT3:  INC     L               ;INCREMENT POINTER
026D'   20 F9                            JR      NZ,LOOP3        ;LOOP TILL DONE WITH PAGE1

026F'   24                               INC     H               ;***DO THE SAME WITH 2ND PAGE***
0270'   BE                       LOOP4:  CP      (HL)
0271'   38 01                            JR      C,DCNT4
0273'   7E                               LD      A,(HL)
0274'   2C                       DCNT4:  INC     L
0275'   20 F9                            JR      NZ,LOOP4
0277'   32 F831                          LD      (BMIN),A        ;SAVE FINAL BMIN VALUE

027A'   C1                               POP     BC              ;RESTORE REGISTERS
027B'   E1                               POP     HL
027C'   F1                               POP     AF
027D'   C9                               RET
                                 PAGE
```

```
                                ;*********************************
                                ;*    BRIGHTNESS ADJUST ROUTINE  - 30 JAN 83     *
                                ;*********************************
                                ;*                                               *
                                ;*  THIS ROUTINE SENDS BRIGHTNESS CORRECTIONS WHEN  *
                                ;*  DBRT AND ABRT DIFFER.                        *
                                ;*                                               *
                                ;*  INPUTS:  DBRT, ABRT, BRTOUT                  *
                                ;*                                               *
                                ;*  OUTPUTS: BRTOUT IS UPDATED                   *
                                ;*                                               *
                                ;*********************************

                                        ORG     0290H

0290'   F5                      BADJ:   PUSH    AF
0291'   C5                              PUSH    BC

0292'   0E 01                           LD      C,OBPORT        ;AIM C AT OUTPUT BRTNSS PORT
0294'   3A F802                         LD      A,(ABRT)        ;LOAD ACTUAL BRIGHTNESS
0297'   47                              LD      B,A
0298'   3A F801                         LD      A,(DBRT)        ;LOAD DESIRED BRIGHTNESS
029B'   90                              SUB     B               ;DBRT - ABRT
029C'   28 0E                           JR      Z,BEXIT         ;DO NOTHING IF THEY MATCH
029E'   3A F803                         LD      A,(BRTOUT)      ;LOAD OLD BRTOUT VALUE
02A1'   38 03                           JR      C,BHI           ;SKIP AHEAD IF ABRT IS TOO HI

02A3'   3C                      BLO:    INC     A               ;KICK OLD BRTOUT UP A TICK
02A4'   18 01                           JR      BOUT            ;SKIP AHEAD

02A6'   3D                      BHI:    DEC     A               ;DROP OLD BRTOUT DOWN A TICK

02A7'   ED 79                   BOUT:   OUT     (C),A           ;OUTPUT CORRECTED VALUE
02A9'   32 F803                         LD      (BRTOUT),A      ;   AND SAVE IT

02AC'   C1                      BEXIT:  POP     BC
02AD'   F1                              POP     AF
02AE'   C9                              RET
                                PAGE
```

```
                                   ;*************************************
                                   ;*    CONTRAST ADJUST ROUTINE  - 30 JAN 83    *
                                   ;*************************************
                                   ;*                                           *
                                   ;*  THIS ROUTINE SENDS CONTRAST CORRECTIONS WHEN  *
                                   ;*  DCON AND ACON DIFFER.                     *
                                   ;*                                           *
                                   ;*  INPUTS:  DCON, ACON, CONOUT               *
                                   ;*                                           *
                                   ;*  OUTPUTS: CONOUT IS UPDATED                *
                                   ;*                                           *
                                   ;*************************************

                                           ORG     02C0H

02C0'   F5                         CADJ:   PUSH    AF
02C1'   C5                                 PUSH    BC

02C2'   0E 02                              LD      C,OCPORT        ;AIM C AT OUTPUT CONTRAST PORT
02C4'   3A F812                            LD      A,(ACON)        ;LOAD ACTUAL CONTRAST
02C7'   47                                 LD      B,A
02C8'   3A F811                            LD      A,(DCON)        ;LOAD DESIRED CONTRAST
02CB'   90                                 SUB     B               ;DCON - ACON
02CC'   28 0E                              JR      Z,CEXIT         ;DO NOTHING IF THEY MATCH
02CE'   3A F813                            LD      A,(CONOUT)      ;LOAD OLD CONOUT VALUE
02D1'   38 03                              JR      C,CHI           ;SKIP AHEAD IF ACON IS TOO HI

02D3'   3C                         CLO:    INC     A               ;KICK OLD CONOUT UP A TICK
02D4'   18 01                              JR      COUT            ;SKIP AHEAD

02D6'   3D                         CHI:    DEC     A               ;DROP OLD CONOUT DOWN A TICK

02D7'   ED 79                      COUT:   OUT     (C),A           ;OUTPUT CORRECTED VALUE
02D9'   32 F813                            LD      (CONOUT),A      ;   AND SAVE IT

02DC'   C1                         CEXIT:  POP     BC
02DD'   F1                                 POP     AF
02DE'   C9                                 RET

                                           END
```

```
Macros:

Symbols:
ABRT    F802    ACON    F812    ADLOOP  01E9'   ADPAGE  01E0'
AVGBRT  0190'   BADJ    0290'   BBIT    0000    BEXIT   02AC'
BHI     02A6'   BLO     02A3'   BMAX    F830    BMIN    F831
BOUT    02A7'   BRTOUT  F803    BUFDAT  0080'   CADJ    02C0'
CBIT    0001    CEXIT   02DC'   CHI     02D6'   CLO     02D3'
CLRMEM  0030'   CNTRST  0200'   CONOUT  F813    CONVRT  0100'
COUT    02D7'   CUPDAT  0080'   DBRT    F801    DBSF    0028
DCNT1   023C'   DCNT2   0244'   DCNT3   026C'   DCNT4   0274'
DCON    F811    DCSF    0064    DIVIDE  0120'   DUNTST  012D'
DVEND   013E'   DVLOOP  0124'   EXEC    0040'   FMAX    0230'
FMIN    0260'   IBPORT  0001    ICPORT  0002    IDPORT  0003
INIT    0000'   ISPORT  0000    ISTAT   F821    LOOP1   0238'
LOOP2   0240'   LOOP3   0268'   LOOP4   0270'   OBPORT  0001
OCPORT  0002    OSPORT  0000    OSTAT   F820    OVRFLW  0131'
PBIT    0002    ROUND   0137'   SCAN    0160'   SCRTCH  F800
SHIFT   019D'   STACK   FC00    STEST   0060'   STEXIT  01CA'
SUMTBL  01B0'   TABLE   F900    TSUM    F832    UDBRT   F800
UDCON   F810    WTBIT   0003


No  Fatal error(s)
```

# APPENDIX E

# DATA DICTIONARY

## ROUTINES

### ADPAGE - PAGE ADDITION ROUTINE

PURPOSE: THIS ROUTINE SUMS THE MEMORY PAGE ADDRESSED BY DE. THE PAGE SUM IS LEFT IN HL.

INPUTS: ADDRESS OF PAGE'S FIRST IN DE

OUTPUTS: SUM OF ALL VALUES IN PAGE IN HL

REGISTERS AFFECTED: HL

### AVGBRT - BRIGHTNESS CALCULATION ROUTINE

PURPOSE: THIS ROUTINE COMPUTES ABRT FROM RETICON DATA.

INPUTS: RETICON DATA

OUTPUTS: ABRT IS CALCULATED & UPDATED
(ABRT = TSUM / 512)

ROUTINES CALLED: SUMTBL

MEMORY LOCATIONS AFFECTED: ABRT, TSUM

### BADJ - BRIGHTNESS ADJUST ROUTINE

PURPOSE: THIS ROUTINE UPDATES BRTOUT WHEN DBRT & ABRT DIFFER

INPUTS: DBRT, ABRT, BRTOUT

OUTPUTS: BRTOUT IS UPDATED

MEMORY LOCATIONS AFFECTED: BRTOUT

### BUPDAT - BRIGHTNESS UPDATE ROUTINE

PURPOSE: THIS ROUTINE UPDATES THE VALUE FOR DBRT. FIRST, UDBRT IS READ FROM IBPORT. THIS BCD VALUE IS CONVERTED TO BINARY. THE RESULT IS THEN SCALED (DBRT = 256 X UDBRT / DBSF).

BFCLR (OS0) IS THEN TOGGLED ON AND OFF TO SIGNAL THAT BRIGHTNESS HAS BEEN UPDATED.

INPUTS: IBPORT IS READ

OUTPUTS: UDBRT AND DBRT ARE UPDATED
BFLCR (OS0) IS TOGGLED ON AND OFF

ROUTINES CALLED: CONVRT, DIVIDE

## CADJ - CONTRAST ADJUST ROUTINE

PURPOSE: THIS ROUTINE UPDATES CONOUT WHEN DCON & ACON DIFFER

INPUTS: DCON, ACON, CONOUT

OUTPUTS: CONOUT IS UPDATED

MEMORY LOCATIONS AFFECTED: CONOUT

## CLRMEM - CLEAR MEMORY ROUTINE

PURPOSE: THIS ROUTINE CLEARS A BLOCK OF MEMORY STARTING WITH THE ADDRESS FOUND IN REGISTER PAIR HL. THE SIZE OF MEMORY BLOCK TO CLEAR MUST BE PLACED IN REGISTER PAIR BC.

INPUTS: STARTING ADDRESS OF BLOCK TO CLEAR IN HL
NUMBER OF BYTES TO CLEAR IN BC

OUTPUTS: CLEARS MEMORY BLOCK ADDRESSED BY HL

REGISTER AFFECTED: HL, BC

MEMORY LOCATIONS AFFECTED: (HL) TO (HL + BC - 1)

## CONTRST - ACTUAL CONTRAST ROUTINE

PURPOSE: THIS ROUTINE COMPUTES ACTUAL CONTRAST FROM THE RETICON DATA. THE ALGORITHM USED IS:

$$ACON = 256 \times (BMAX - BMIN) / (2 \times BAVG)$$

INPUTS: ABRT (=BAVG), RETICON DATA

OUTPUTS: ACON, BMAX, BMIN ARE UPDATED

ROUTINES CALLED: FMAX, FMIN, DIVIDE

MEMORY LOCATIONS AFFECTED: BMAX, BMIN, ACON

CONVRT - BCD TO BINARY CONVERSION ROUTINE

PURPOSE: THIS ROUTINES CONVERTS A 2-DIGIT BCD NUMBER TO ITS 8-BIT BINARY EQUIVALENT

INPUTS: 2-DIGIT BCD NUMBER IN A

OUTPUTS: 8-BIT BINARY EQUIVALENT IN A

REGISTERS AFFECTED: A

CUPDAT - CONTRAST UPDATE ROUTINE

PURPOSE: THIS ROUTINE UPDATES THE VALUE FOR DCON. FIRST, UDCON IS READ FROM ICPORT. THIS BCD VALUE IS CONVERTED TO BINARY. THE RESULT IS THEN SCALED (DCON = 256 X UDCON / DCSF). CFCLR (OS1) IS THEN TOGGLED ON AND OFF TO SIGNAL THAT CONTRAST HAS BEEN UPDATED.

INPUTS: ICPORT IS READ

OUTPUTS: UDCON AND DCON ARE UPDATED
CFLCR (OS1) IS TOGGLED ON AND OFF

ROUTINES CALLED: CONVRT, DIVIDE

DIVIDE - DIVIDE ROUTINE

PURPOSE: THIS ROUTINE DIVIDES A 16-BIT UNSIGNED DIVIDEND IN HL BY AN 8-BIT UNSIGNED DIVISOR IN C TO YIELD AN 8-BIT QUOTIENT (WITH LSB ROUNDED) IN L. THE REMAINDER THAT EXISTED BEFORE ROUNDING IS LEFT IN H.

INPUTS: 16-BIT DIVIDEND (DVD) IN HL
8-BIT DIVISOR (DVS) IN C

OUTPUTS: 8-BIT ROUNDED QUOTIENT (QUO) IN L
8-BIT REMAINDER (BEFORE ROUNDING) IN H

REGISTERS AFFECTED: HL

EXEC - EXECUTIVE ROUTINE

PURPOSE: THIS IS THE MAIN PROGRAM. ONCE ENTERED, THIS ROUTINE REPEATS ENDLESSLY, CALLING THE OTHER ROUTINES IN PROPER ORDER.

INPUTS: NONE

OUTPUTS: SEE INDIVIDUAL ROUTINES REFERENCED

ROUTINES CALLED: STEST, SCAN, AVGBRT, CNTRST, BADJ, CADJ

FMAX - FIND MAX ROUTINE

PURPOSE: THIS ROUTINE FINDS THE MAX VALUE IN THE RETICON DATA TABLE

INPUTS: RETICON DATA

OUTPUTS: BMAX IS UPDATED

MEMORY LOCATIONS AFFECTED: BMAX

FMIN - FIND MIN ROUTINE

PURPOSE: THIS ROUTINE FINDS THE MIN VALUE IN THE RETICON DATA TABLE

INPUTS: RETICON DATA

OUTPUTS: BMIN IS UPDATED

MEMORY LOCATIONS AFFECTED: BMIN

INIT - INITIALIZATION ROUTINE

PURPOSE: THIS ROUTINE INITIALIZES THE CONTROLLER WHEN THE SYSTEM FIRST COMES UP OR IS INITIALIZED. RAM WORK AREAS ARE CLEARED. USER INTERFACE CIRCUITS ARE RESET. DBRT AND DCON ARE PRESET TO DEFAULT VALUES SET BY SWITCHES S5 AND S6.

INPUTS: NONE

OUTPUTS: STACK POINTER IS INITIALIZED
RAM WORK AREAS ARE CLEARED
USER INTERFACE CIRCUITS ARE RESET
DBRT AND DCON ARE INITIALIZED

ROUTINES CALLED: CLRMEM, BUPDAT, DUPDAT

SCAN - RETICON SCAN ROUTINE

PURPOSE: THIS ROUTINE READS 512 BYTES OF DATA FROM THE IDPORT.

INPUTS: RETICON SAMPLES ARE READ FROM IDPORT

OUTPUTS: 512-BYTE RETICON DATA TABLE IS UPDATED
RSCAN (OS3) IS TOGGLED ON & OFF

STEST - STATUS UPDATE ROUTINE

PURPOSE: THIS ROUTINE UPDATES STORED VALUES FOR DBRT or DCON, IF BUFLG OR CUFLG (RESPECTIVELY) ARE SET.

INPUTS: INPUT STATUS PORT IS READ

OUTPUTS: DBRT AND DCON ARE UPDATED, IF UPDATES NEEDED

ROUTINES CALLED: BUPDAT, CUPDAT

SUMTBL - TABLE SUMMING ROUTINE

PURPOSE: THIS ROUTINE SUMS ALL 512 BYTES STORED IN THE RETICON DATA TABLE AND SAVES RESULT IN TSUM.

INPUTS: RETICON DATA TABLE

OUTPUTS: TSUM IS UPDATED

ROUTINES CALLED: ADPAGE

MEMORY LOCATIONS AFFECTED: TSUM (3 BYTES)

## VARIABLES

ABRT - ACTUAL BRIGHTNESS VALUE

ABRT IS THE 8-BIT AVERAGE OF ALL 512 RETICON SAMPLES STORED IN THE DATA TABLE. ALSO KNOWN AS BAVG.

COMPUTED AS: TSUM / 512

COMPUTED BY: AVGBRT

USED BY: CONTRST, BADJ

ACON - ACTUAL CONTRAST VALUE

ACON IS THE 8-BIT VALUE FOR ACTUAL CONTRAST COMPUTED FROM THE RETICON DATA SAMPLES. NOTE THAT ACON IS ACTUALLY SCALED BY 256 SINCE TRUE ACON WOULD BE A FRACTION.

COMPUTED AS: ACON = 256 X (BMAX - BMIN) / (2 X BAVG)

COMPUTED BY: CONTRST

USED BY: CADJ

BAVG - AVERAGE BRIGHTNESS VALUE

SAME AS ABRT (SEE LISTING FOR ABRT)

BMIN - MINIMUM BRIGHTNESS VALUE

BMIN IS THE MINIMUM VALUE FOUND IN THE RETICON DATA TABLE. THIS TERM IS USED IN COMPUTING ACON.

UPDATED BY: FMIN

USED BY: CNTRST

BMAX - MAXIMUM BRIGHTNESS VALUE

BMAX IS THE MAXIMUM VALUE FOUND IN THE RETICON DATA TABLE. THIS TERM IS USED IN COMPUTING ACON.

UPDATED BY: FMAX

USED BY: CNTRST

BRTOUT - BRIGHTNESS OUTPUT VALUE

BRTOUT IS THE 8-BIT VALUE WHICH CONTROLS THE BRIGHTNESS OUTPUT CIRCUITRY. BRTOUT IS ITERATIVELY ADJUSTED UP OR DOWN BY BADJ TO COMPENSATE FOR DIFFERENCES BETWEEN DBRT AND ABRT.

SET BY: BADJ

SENT TO: OBPORT (OUTPUT PORT 1)

CONOUT - CONTRAST OUTPUT VALUE

CONOUT IS THE 8-BIT VALUE WHICH CONTROLS THE CONTRAST OUTPUT CIRCUITRY. CONOUT IS ITERATIVELY ADJUSTED UP OR DOWN BY CADJ TO COMPENSATE FOR DIFFERENCES BETWEEN DCON AND ACON.

SET BY: CADJ

SENT TO: OCPORT (OUTPUT PORT 2)

DBRT - DESIRED BRIGHTNESS VALUE

DBRT IS THE 8-BIT SCALED VALUE FOR DESIRED BRIGHTNESS. THIS VARIABLE IS SCALED SO THAT DBRT CAN BE COMPARED DIRECTLY WITH ABRT.

COMPUTED AS: DBRT = 256 X UDBRT / DBSF

COMPUTED BY: BUPDAT

USED BY: BADJ

DCON - DESIRED CONTRAST VALUE

DCON IS THE 8-BIT SCALED VALUE FOR DESIRED CONTRAST. THIS VARIABLE IS SCALED SO THAT DCON CAN BE COMPARED DIRECTLY WITH ACON.

COMPUTED AS: DCON = 256 X UDCON / DCSF

COMPUTED BY: CUPDAT

USED BY: CADJ

ISTAT - INPUT STATUS BYTE

ISTAT IS THE INPUT STATUS BYTE READ AT THE ISPORT. THIS BYTE IS USED BY THE Z80 TO DETERMINE THE STATUS OF THE INTERFACE CIRCUITS. THE STATES OF BITS IS0 AND IS1 DETERMINE WHETHER OR NOT DBRT AND DCON ARE TO BE UPDATED.

UPDATED BY: STEST

USED BY: STEST

OSTAT - OUTPUT STATUS BYTE

OSTAT IS THE OUTPUT STATUS BYTE SENT TO THE INTERFACE AND ADC CIRCUITS VIA THE OSPORT. THE INDIVIDUAL BITS SERVE TO TRIGGER SPECIFIC ACTIONS IN THE CONTROLLER HARDWARE. OS0 AND OS1 RESET BUFLG AND CUFLG (RESPECTIVELY). OS2 PRESETS COUNTERS TO DEFAULT VALUES. OS3 SIGNALS THE WSC THAT THE Z80 WILL READ THE NEXT RETICON SCAN.

AFFECTED BY: INIT (NOTE THAT THE STORED VALUE FOR OSTAT IS UNAFFECTED BY OTHER ROUTINES; BUT, SPECIFIC BITS SENT TO THE OSPORT ARE MOMENTARILY TOGGLED ON AND OFF BY ROUTINES INIT, BUPDAT, CUPDAT & SCAN.)

USED BY: INIT, BUPDAT, CUPDAT, SCAN

TSUM - RETICON TABLE SUM

TSUM IS THE SUM OF ALL VALUES STORED IN THE RETICON DATA TABLE. THIS TERM IS USED IN COMPUTING ABRT.

COMPUTED BY: SUMTBL

USED BY: AVGBRT

UDBRT - UNSCALED DESIRED BRIGHTNESS VALUE

UDBRT IS THE 8-BIT BCD VALUE FOR DESIRED BRIGHTNESS ENTERED BY THE USER VIA THE INTERFACE CIRCUITS. THIS VALUE MUST BE CONVERTED TO BINARY AND SCALED BEFORE A COMPARISON CAN BE MADE WITH ABRT.

READ FROM: IBPORT (INPUT PORT 1)

USED BY: BUPDAT

UDCON - UNSCALED DESIRED CONTRAST VALUE

UDCON IS THE 8-BIT BCD VALUE FOR DESIRED CONTRAST ENTERED BY THE USER VIA THE INTERFACE CIRCUITS. THIS VALUE MUST BE CONVERTED TO BINARY AND SCALED BEFORE A COMPARISON CAN BE MADE WITH ACON.

READ FROM: ICPORT (INPUT PORT 2)

USED BY: CUPDAT

MISCELLANEOUS

BFCLR - BRIGHTNESS FLAG CLEAR BIT (OS0)

BFCLR RESETS THE BRIGHTNESS FLAGS (BUFLG AND BNUFLG) AFTER DBRT HAS BEEN UPDATED.

BUFLG - BRIGHTNESS UPDATE FLAG (IS0)

THIS FLAG SIGNALS THE Z80 THAT THE USER WISHES TO UPDATE DBRT.

CFLCR - CONTRAST FLAG CLEAR BIT (OS1)

CFCLR RESETS THE CONTRAST FLAGS (CUFLG AND CNUFLG) AFTER DCON HAS BEEN UPDATED.

CUFLG - CONTRAST UPDATE FLAG (IS1)

THIS FLAG SIGNALS THE Z80 THAT THE USER WISHES TO UPDATE DBRT.

DBSF - DESIRED BRIGHTNESS SCALE FACTOR

DBSF IS THE FACTOR USED IN SCALING DBRT FOR DIRECT COMPARISON WITH ABRT. DBSF = 40 (H).

DCSF - DESIRED CONTRAST SCALE FACTOR

DCSF IS THE FACTOR USED IN SCALING DCON FOR DIRECT COMPARISON WITH ACON. DCSF = 100 (H).

IBPORT - INPUT BRIGHTNESS PORT (IN PORT 1)

IBPORT IS USED TO INPUT UDBRT FROM THE USER INTERFACE CIRCUITS.

ICPORT - INPUT CONTRAST PORT (IN PORT 2)

ICPORT IS USED TO INPUT UDCON FROM THE USER INTERFACE CIRCUITS

IDPORT - INPUT DATA PORT (IN PORT 3)

IDPORT IS USED TO INPUT RETICON DATA SAMPLES FROM THE ADC CIRCUIT.

ISPORT - INPUT STATUS PORT (IN PORT 0)

ISPORT IS USED TO INPUT STATUS FLAGS FROM THE USER INTERFACE CIRCUITS.

OBPORT - OUTPUT BRIGHTNESS PORT (OUT PORT 1)

OBPORT IS USED TO SEND BRTOUT VALUES TO THE BRIGHTNESS DAC.

OCPORT - OUTPUT CONTRAST PORT (OUT PORT 2)

OCPORT IS USED TO SEND CONOUT VALUES TO THE CONTRAST DAC.

OSPORT - OUTPUT STATUS PORT (OUT PORT 0)

OSPORT IS USED TO SEND OUTPUT CONTROL SIGNALS TO THE USER INTERFACE AND ADC CIRCUITS.

PRESET - PRESET BIT (OS2)

PRESET INITIALIZES COUNTERS IN THE USER INTERFACE CIRCUITS TO DEFAULT VALUES FOR DBRT & DCON WHEN THE SYSTEM IS RESET.

RSCAN - RETICON SCAN BIT (OS3)

RSCAN RESETS THE WAIT STATE CONTROLLER WHEN THE Z80 IS READY TO READ A RETICON SCAN.

SCRTCH - SCRATCHPAD WORK AREA

SCRTCH IS THE STARTING ADDRESS OF THE SCRATCHPAD WORK AREA USED FOR INTERMEDIATE STORAGE OF DATA VALUES.
SCRTCH = F800 (H)

STACK

STACK IS THE TOP OF THE USER'S STACK AREA.
STACK = FC00 (H)

TABLE

TABLE IS THE STARTING ADDRESS OF THE 512-BYTE RETICON DATA TABLE. TABLE = F900 (H)

# VITA

Barry Douglas Baxley was born on 7 April 1952 in Dallas, Texas. He graduated from Coronado High School in Scottsdale, Arizona in May 1970. After two years of study at Mesa Community College, Mesa, Arizona he entered the United States Air Force. In December 1976, he graduated Summa Cum Laude from Arizona State University, Tempe, Arizona with the degree of Bachelor of Science in Engineering (majoring in Electrical Engineering). He proceeded to Officer Training School, where he was commissioned in March 1977. He was then assigned to the Avionics Laboratory at Wright-Patterson AFB, where he served three years as an Avionic Systems Engineer, working in the areas of Automatic Test Equipment and Fault Tolerant Computing. During his fourth year in the Avionics Laboratory, he served as a Staff Avionics Engineer, and finally as Acting Chief, Avionics Laboratory Programs Group. In June 1981, Captain Baxley entered the School of Engineering, Air Force Institute of Technology. He is a member of the Eta Kappa Nu, Tau Beta Pi, and Phi Kappa Phi honor societies.

Permanent Address: 8341 East Indian School Road
Scottsdale, Arizona 85251

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
AFIT/GE/EE/83M-1

2. GOVT ACCESSION NO.
A135871

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
A Sine-Wave Grating Controller For Vision Testing

5. TYPE OF REPORT & PERIOD COVERED
MS THESIS

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Barry D. Baxley
Captain USAF

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Air Force Institute of Technology (AFIT/EN)
Wright-Patterson AFB, Ohio 45433

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Aerospace Medical Research Laboratory (AFAMRL/HEA)
Wright-Patterson AFB, Ohio 45433

12. REPORT DATE
March 1983

13. NUMBER OF PAGES
154

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES
Approved for public release: IAW AFR 190-17.
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433
1 2 SEP 1983

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

| | |
|---|---|
| Sine-Wave Grating | Luminance Control |
| Vision Testing | Contrast Control |
| Contrast Sensitivity | Microprocessor |

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report summarizes design, development and testing of a microprocessor-based system that controls brightness and contrast levels of video monitors. This controller is to be used by scientists in maintaining calibration of video monitors during vision research.

Prototype hardware for a video controller was designed and built. The controller requires a sine-wave grating to be displayed on a cathode ray tube (CRT). Luminance samples are measured directly from the CRT via a 512-element

photodiode array. Analog information from the array is digitized and stored in a Z80 microcomputer. The sampled data are then used to compute CRT brightness and contrast. Computed brightness and contrast values are compared with desired values, and corrections are made to brightness and contrast c ontrol circuits for the video monitor.

Most of the hardware required for the controller was built and tested. Hardware completed included the video sampling and digitizing circuits, as well as Z80 input/output and user interface circuits. In addition, all of the controller software was written and tested.

Additional development is required before a working controller can be demonstrated. Brightness and contrast control circuits must be built and tested. Anomolies in performance of the commercially-procured video sampling circuits should be examined. Finally, closed-loop system operation must be tested and analyzed.

DATE

ILME